MANUAL DE PROBLEMAS COMPETENCIAS DE PROGRAMACIÓN

INSTITUCIONES PÚBLICAS Y PRIVADAS DE PUERTO RICO

PROF. NELLIUD D. TORRES

AGRADECIMIENTOS

INTRODUCCIÓN

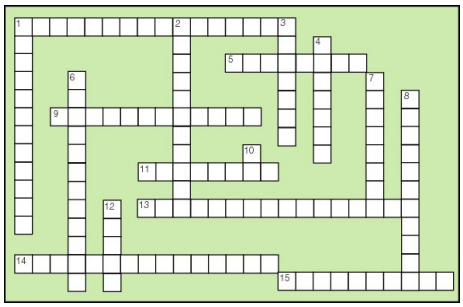
TABLA DE CONTENIDO

UNIVERSIDAD DE PUERTO RICO

EN BAYAMÓN

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Principiantes	**Universidad**: UPR – Bayamón
Autor: Nelliud D. Torres	**Tipo de competencia**: Programación
Problema #: 1	

LA PALABRA CRUZADA



Desarrolle un programa que pida por pantalla una palabra de tres (3) a 13 caracteres. Con esa palabra s e va a formar una X en donde el caracter del medio se repite una sola vez. El programa debe validar que la cantidad de caracteres entrados sea impar y que no sea menor de 3 caracteres ni mayor de 13. Si va a escribir el programa en un lenguaje de orientación gráfica como Visual Basic, asegúrese de poner el tipo de letra en la salida como `Courier New`.

Ejemplo 1:

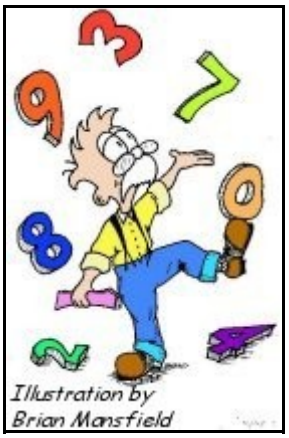
```
Entre una palabra impar de 3 a 13 caracteres: el
Palabra menor de 3 caracteres, trate de nuevo

Entre una palabra impar de 3 a 13 caracteres: amor
Palabra par, trate de nuevo

Entre una palabra impar de 3 a 12 caracteres: parangutirimicuaro
Palabra mayor de 13 caracteres, trate de nuevo

Entre una palabra impar de 3 a 13 caracteres: linux

l   l
 i i
  n
 u u
x   x
```

Ejemplo 2:

```
Entre una palabra de 3 a 12 caracteres: Microsoft

M       M
 i     i
  c   c
   r r
    o
   s s
  o   o
 f     f
t       t
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Principiantes	**Universidad**: UPR – Bayamón
Autor: Nelliud D. Torres	**Tipo de competencia**: Programación
Problema #: 2	

Números Pseudoparásitos



Los números parásitos (según el Dr. Googol) son aquellos números que al multiplicarse por un número de un dígito, cambia el dígito de la última posición a la primera. En otras palabras el multiplicando es similar al resultado excepto que el último dígito es el primero del resultado. El siguiente ejemplo se explica por si solo: 102,56**4** X **4** = **4**10,256. Para que sea un verdadero número parásito el multiplicador debe ser similar al número que cambia de posición en el resultado. Lamentablemente son muy escasos estos números. Una variación son los pseudoparásitos que al multiplicarse por 4 cambian el último dígito al principio, pero este no es similar al multiplicador. Un ejemplo es: 153,84**6** X 4 = **6**15,384. Aunque estos otros números son también raros, ocurren con más frecuencia que los números parásitos especialmente cuando el multiplicador es 4. Escriba un programa que busque aquellos números parásitos de 6 dígitos (100,000 – 999,999) que al multiplicarse por 4 cambie el último dígito de lugar.

Input

El programa no va a pedir absolutamente nada al usuario. .

Output (screen)

Simplemente va a mostrar en pantalla los resultados encontrados y va a notificar la cantidad de números encontrados.

Corrida de ejemplo

```
128,205 X 4 = 512,820
```

-
-

```
Total de números pseudoparásitos de 6 dígitos(*4): 99
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Principiantes	**Universidad**: UPR – Bayamón
Autor: Antonio Huertas	**Tipo de competencia**: Programación
Problema #: 3	

MI VERSIÓN DE TAIL

El sistema operativo UNIX provee un comando llamado **tail** que muestra las últimas **n** líneas de un archivo de texto. Escriba un programa que pregunte el nombre de un archivo de texto y un número y que se comporte como **tail**. Si el archivo tiene menos de **n** líneas, el programa debe mostrar *todo* el contenido del archivo. Asuma que cada línea del archivo acaba en '\n'. Valide que el archivo exista y que el valor de **n** sea 0 ó más.

Archivo de Prueba (kernel.txt):

```
Que es el kernel?

El kernel o núcleo del sistema operativo es el programa que se comunica
directamente con el hardware. Esta es la parte del sistema operativo que
se carga en RAM cuando se enciende la computadora y permanece en RAM hasta
que la computadora se apaga. Está escrito, en el caso de Unix, mayormente
en C con un poco de lenguaje de ensamblaje. El kernel debe interactuar con
los usuarios, con los programas y, obviamente, con el hardware.
```

Ejemplo:

```
Indique el nombre del archivo: abc.txt
El archivo no existe, trate de nuevo.

Indique el nombre del archivo: kernel.txt
Indique la cantidad de líneas: -10
La cantidad de líneas es menor de 0, trate de nuevo.

Indique la cantidad de líneas: 4

Las últimas 4 líneas de kernel.txt son:

se carga en RAM cuando se enciende la computadora y permanece en RAM hasta
que la computadora se apaga. Está escrito, en el caso de Unix, mayormente
en C con un poco de lenguaje de ensamblaje. El kernel debe interactuar con
los usuarios, con los programas y, obviamente, con el hardware.
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Principiantes	**Universidad**: UPR – Bayamón
Autor: Antonio Huertas	**Tipo de competencia**: Programación
Problema #: 4	

MI VERSIÓN DE CMP

El sistema operativo UNIX provee un comando llamado **cmp** que dos archivos y que determina si son idénticos o no. Si los archivos son idénticos, el comando muestra un mensaje que lo indica. Si no lo son, comando muestra el número de línea y de caracter donde aparece la primera diferencia. Escriba un programa que pregunte el nombre de dos archivos de texto y que se comporte como **cmp**. Asuma que cada línea del archivo acaba en '\n'. Valide que los archivos existan y que se comparen dos archivos con nombres diferentes.

Archivo de Prueba 1 (ferreteria.txt):

```
222 Martillo 2.50 10
444 Serrucho 10.00 3
111 Clavos 1.00 15
333 Pala 4.00 5
777 Tornillos 1.00 20
555 Destornillador 3.00 4
```

Archivo de Prueba 2 (ferreteria2.txt):

```
222 Martillo 2.50 10
444 Brocha 10.00 3
111 Pintura 1.00 15
777 Tornillos 1.00 20
888 Cadena 7.00 1
555 Destornillador 3.00 4
```

Ejemplo:

```
Indique el nombre del archivo #1: abc.txt
El archivo #1 no existe, trate de nuevo.

Indique el nombre del archivo #1: ferreteria.txt
Indique el nombre del archivo #2: abc.txt
El archivo #2 no existe, trate de nuevo.

Indique el nombre del archivo #2: ferreteria.txt
Los nombres de los archivos son iguales, trate de nuevo.

Indique el nombre del archivo #2: ferreteria2.txt

Los archivos ferretería.txt y ferreteria2.txt no son iguales.
La primera diferencia está en la línea #2, caracter #5.
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Principiantes	**Universidad**: UPR – Bayamón
Autor: Antonio Huertas	**Tipo de competencia**: Programación
Problema #: 5	

JUGANDO CON FECHAS

Escriba un programa que pregunte una fecha y que muestre la fecha del próximo día. La fecha deberá ser entrada en formato **mm/dd/aaaa**, donde **mm** es el mes, **dd** es el día y **aaaa** es el año. El programa deberá validar la fecha:

- El año deberá estar entre 1 y 2100.
- El mes deberá estar entre 1 y 12.
- El día deberá estar entre:
  - 1 y 30 para los meses 4 (abril), 6 (junio), 9 (septiembre) y 11 (noviembre)
  - 1 y 31 para los meses 1 (enero), 3 (marzo), 5 (mayo), 7 (julio), 8 (agosto), 10 (octubre) y 12 (diciembre)
  - 1 y 28 para el mes 2 (febrero) si el año no es bisiesto; 1 y 29 para el mes 2 (febrero) si el año es bisiesto

Ejemplo 1:

```
Indique la fecha: 11/31/2006
La fecha es incorrecta, trate de nuevo.

Indique la fecha: -11/30/2006
La fecha es incorrecta, trate de nuevo.

Indique la fecha: 11/30/2006
La fecha del próximo día es 12/1/2006.
```

Ejemplo 2:

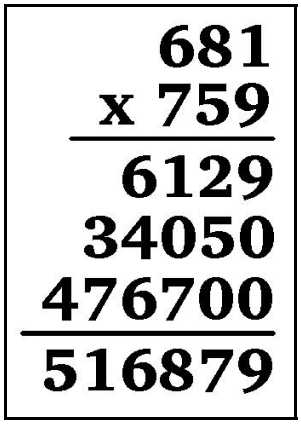
```
Indique la fecha: 2/28/2006
La fecha del próximo día es 3/1/2006.
```

Ejemplo 3:

```
Indique la fecha: 2/28/2008
La fecha del próximo día es 2/29/2008.
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Intermedio	**Universidad**: UPR – Bayamón
Autor: Nelliud D. Torres	**Tipo de competencia**: Programación
Problema #: 1	

Números Vampiros



Los números vampiros son producto de dos números progenitores que cuando se multiplican, se mezclan con el resultado. Por ejemplo la siguiente multiplicación produce un número vampiro: 27 X 81 = 2187. Esto se debe a que los dígitos 2, 7, 8 y 1 se encuentran tanto en el resultado como en los números progenitores. Otro ejemplo puede ser 1,435 el cual es el resultado de 35 X 41. Un verdadero número vampiro cumple los siguientes requisitos:

1. Tienen una cantidad par de dígitos.

2. Cada uno de los números progenitores tiene la mitad de los números del resultado.

3. Un verdadero número vampiro no se crea al incluírsele ceros al final. Por ejemplo 270,000 X 810,000 = 218,700,000,000 no es un verdadero número vampiro

Haga un programa que calcule los verdaderos números vampiros de 4 (progenitores = 2 dígitos), 6 (progenitores = 3 dígitos) y 8 (progenitores = 4 dígitos) dígitos.

Input (screen)

El programa va a pedir por pantalla la cantidad de dígitos que quiere cotejar y sólo ofrecerá tres alternativas 4, 6 y 8.

Output (screen & file:vampiro.out)

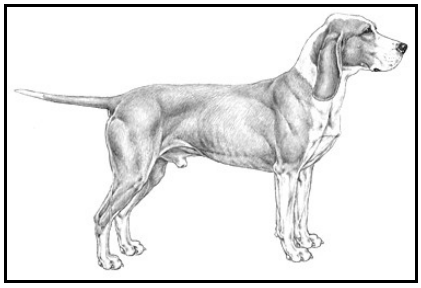
El programa va mostrar en pantalla y en archivo el resultado según lo solicitó el juez. No solamente va a mostrar la cantidad de números vampiros, sino que también va a mostrar el total encontrado.

Corrida de ejemplo

```
Indique la cantidad de dígitos(4,6,8): 2
Cantidad indicada incorrecta, trate de nuevo

Indique la cantidad de dígitos(4,6,8): 4

15 x 93 = 1395
21 x 60 = 1260
21 x 87 = 1827
27 x 81 = 2187
30 x 51 = 1530
35 x 41 = 1435
80 x 86 = 6880


Total de números vampiros de 4 dígitos es: 7
```

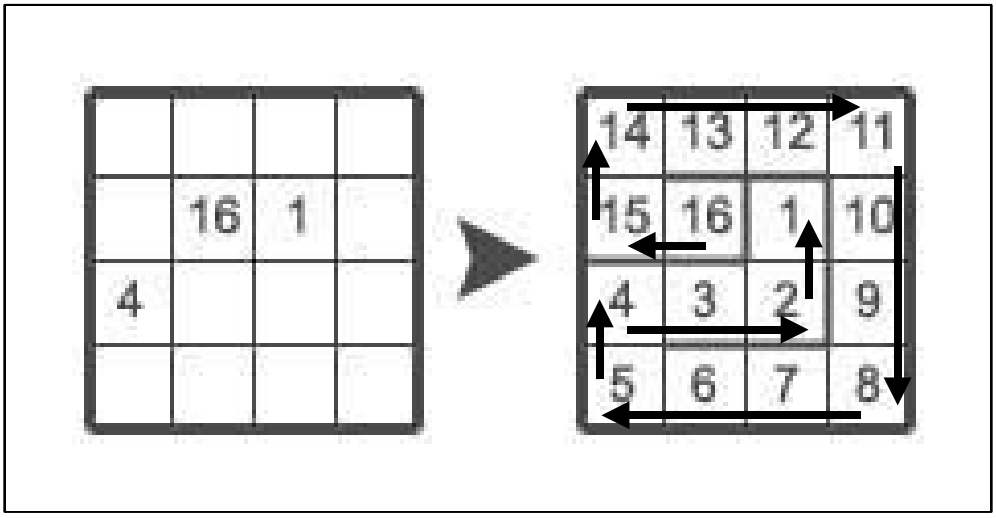
Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Intermedio	**Universidad**: UPR – Bayamón
Autor: Nelliud D. Torres	**Tipo de competencia**: Programación
Problema #: 2	

El Sabueso



Un sabueso ha recorrido completamente un tablero avanzando de una casilla a otra vecina en horizontal o vertical (nunca en diagonal) sin pasar dos veces por la misma casilla y sin dejar ninguna sin visitar. El recorrido será del número mayor hacia el número menor (no necesariamente tiene que terminar en 1) hasta que se llenen todos los encasillados.

Un ejemplo de un tablero de 4 X 4 es:



Para simplificar un poco el problema, se va a indicar la posición inicial en donde se va a comenzar el recorrido, el número inicial (mayor) y el tamaño del tablero va a ser fijo (6 X 6).

Input (file:sabueso.in)

El programa va a leer un archivo en donde la primera línea tiene la cantidad de tableros que va a leer. La próxima línea tiene la posición inicial dentro del tablero en donde se va a comenzar el recorrido (base 1). La tercera línea va a tener el número con el cual se va a comenzar el recorrido y luego sigue el tablero de 6 X 6 en donde cada posición estará representada por un punto menos aquellos encasillados que tengan ya el número definido. Cada número o punto estará separado por un espacio en blanco. Si hay más de un tablero, habrá una línea que separe un tablero de otro seguido de la posición inicial del tablero y en la próxima línea el número inicial.

Output (file:sabueso.out)

El programa guardará en un archivo cada tablero con sus correspondientes soluciones. Es importa indicar que para facilidad de lectura de los resultados por parte de los jueces, aquellos números entre 0 a 9 deben tener un espacio adicional al frente para que quede el tablero completamente cuadrado. Aquella pareja que someta un resultado que no incluya este formato, sacará un *incorrect output* y la penalidad de tiempo que se indique en las reglas de las competencias.

Test Data Input:

```
1
3 3
37
. . . . . 22
. . . 33 . .
. 30 . . . .
4 . . . 8 .
. 12 . . . .
. . . . . 17
```

Test Data Output:

```
27 26 25 24 23 22
28 31 32 33 34 21
29 30 37 36 35 20
 4  5  6  7  8 19
 3 12 11 10  9 18
 2 13 14 15 16 17
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Intermedio	**Universidad**: UPR – Bayamón
Autor: Nelliud D. Torres	**Tipo de competencia**: Programación
Problema #: 3	

SOLON File System – All File Size

Definición del problema

El sistema operativo VSD-Ultrix Vista utiliza el sistema de archivos SOLON. Este sistema utiliza una tabla de direcciones de archivos (File Allocation Table) de 32 bits llamada el SOLON-FAT. Un ejemplo del SOLON-FAT sigue a continuación:

ID	Filename	FC	In-link	Start-Addr	End-Addr
1	aoki.gif	1	5	0100	0126
2	ethereal.txt	1	7	2003	2016
3	winident.dll	1	8	0033	0053
4	aoki.gif	0	0	0054	0080
5	aoki.gif	0	4	0500	1024
6	winident.dll	0	0	0271	0460
7	ethereal.txt	0	0	0001	0032
8	winident.dll	0	6	0127	0200

El SOLON-FAT se compone de lo siguiente:

- Cada entrada en el SOLON-FAT está identificada con un número natural (1,2,3…) y ese es su ID dentro de la tabla. (Siempre comienza en 1).
- Hay un campo *binario (0,1)* que identifica el principio del archivo. Este campo se le llama el *First-Chain* (FC).
- Hay un campo llamado *in-link* que identifica la posición dentro del SOLON-FAT del próximo segmento de datos referente al archivo definido en el campo de *filename*. Si este campo está en 0 significa que éste es el último segmento del archivo.
- Los campos de *Start-Addr* and *End-Addr* identifican donde comienza y termina ese segmento de datos referente al archivo descrito en *filename*.

Por ejemplo el archivo `aoki.gif` tiene 3 entradas en el SOLON-FAT (IDS 1, 5 y 4). El archivo esta dividido en 3 segmentos. El primer segmento comienza en la posicion 100 y termina en la 126, el segundo segmento comienza en la pos. 500 y termina en la pos. 1024, y el ultimo segmento comienza en la posicion 54 y termina en la 80.

El programa

Usted deberá construir un programa que dado un archivo SFAT.TXT pueda calcular el tamaño de todos los archivos contenidos en el SFAT.

Ejemplo: Dado un SFAT.TXT como el presentado anteriormente su sistema desplegará en pantalla:

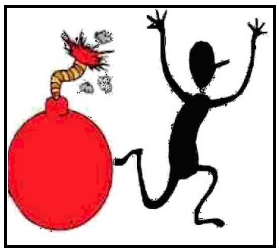
```
aoki.gif 578 bytes 3 segmentos
ethereal.txt 57 bytes 2 segmentos
winident.dll 306 bytes 3 segmentos
```

IMPORTANTE:

- El primer segmento de todos los archivos contenidos en el SOLON-FAT están en las primeras posiciones.
- Un archivo puede estar guardado en 1 o más segmentos.
- Si un segmento empieza en la posición 1 y termina en la 3 es de tamaño 3 y **NO** de tamaño (3-1)=2. Esto es cierto ya que el *byte* 1, 2 y 3 pertenecen al mismo segmento.

Para este problema el número mayor de entradas en el SOLON-FAT **no excederá nunca** 20.

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Intermedio	**Universidad**: UPR – Bayamón
Autor: ACM	**Tipo de competencia**: Programación
Problema #: 4	



BOMBAS

El juego *Minesweeper* se basa en ir adivinando en donde se encuentran las bombas ocultas. Para que el jugador pueda determinar en donde se encuentran, utiliza de referencia los números que le van indicando la proximidad de las bombas. De la misma forma vamos a analizar los números de un tablero para determinar en donde se encuentran las diferentes bombas. Cada número indica cuántas bombas hay en las casillas vecinas, en horizontal, vertical y diagonal. Ninguna casilla lleva más de una bomba y donde hay número no hay bomba. Haga un programa que lea de un archivo una serie de tableros y guarde los resultados en otro archivo.

Input

El archivo comenzará con una primera línea que indicará el numero de tableros que se van a evaluar. El próximo record debe contener la cantidad de filas y columnas que contiene el tablero (base 1). Luego debe seguir el tablero. En aquellos encasillados que no lleva número, se va aponer un punto (.). Cada posición va a estar separada de un espacio en blanco. Cada tablero va a tener sus dimensiones y va a estar separado de la otra tabla por una línea en blanco.

Output

La salida consiste en mostrar el tablero con sus correspondientes bombas. Las bombas se van a dibujar utilizando el asterisco (*). En donde no hay bombas, se deja el punto. El programa debe validar cualquier posible error que pueda surgir de leer los datos. Al final se debe indicar el total de bombas encontradas en el tablero.

Sample Input (File: bombas.in)

```
1
4 4
. . 2 .
1 2 . .
. . . 4
. . . .
```

Sample Output (File:bombas.out)

```
. * 2 .
1 2 . *
. . * 4
. . * *
Total de bombas: 5
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Intermedios	**Universidad**: UPR – Bayamón
Autor: Antonio Huertas	**Tipo de competencia**: Programación
Problema #: 5	

FECHAS DEL FUTURO

Escriba un programa que pregunte una fecha y un número positivo **n** y que muestre la fecha de **n** días en el futuro. La fecha deberá ser entrada en formato **mm/dd/aaaa**, donde **mm** es el mes, **dd** es el día y **aaaa** es el año. El programa deberá validar la fecha:

- El año deberá estar entre 1 y 2100.
- El mes deberá estar entre 1 y 12.
- El día deberá estar entre:
  - 1 y 30 para los meses 4 (abril), 6 (junio), 9 (septiembre) y 11 (noviembre)
  - 1 y 31 para los meses 1 (enero), 3 (marzo), 5 (mayo), 7 (julio), 8 (agosto), 10 (octubre) y 12 (diciembre)
  - 1 y 28 para el mes 2 (febrero) si el año no es bisiesto; 1 y 29 para el mes 2 (febrero) si el año es bisiesto

Ejemplo 1:

```
Indique la fecha: 11/31/2006
El día es incorrecto para el mes indicado, trate de nuevo.

Indique la fecha: -11/25/2006
El mes es incorrecto, trate de nuevo.

Indique la fecha: 11/25/2106
El año es incorrecto, trate de nuevo.

Indique la fecha: 11/25/2006
Indique el número: -8
El número es no es positivo, trate de nuevo.

Indique la fecha: 11/25/2006
Indique el número: 8
La fecha 8 días en el futuro es 12/3/2006
```

Ejemplo 2:

```
Indique la fecha: 2/28/2006
Indique el número: 4
La fecha 4 días en el futuro es 3/4/2006.
```

Ejemplo 3:

```
Indique la fecha: 2/28/2008
Indique el número: 4
La fecha 4 días en el futuro es 3/3/2008.
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Experto	**Universidad**: UPR – Bayamón
Autor: Nelliud D. Torres	**Tipo de competencia**: Programación
Problema #: 1	

Klingon Paths

En la serie de *Star Trek* la raza kingloniana (*klingon*) se caracteriza por ser grandes guerreros y no tener miedo a la muerte. Durante una de sus múltiples guerras, se encontraron con un enemigo que registra la localización de cada nave según entra en cada cuadrante. Si la nave vuelve al mismo lugar, se activa una bomba que destruye el cuadrante completo. Como los *klingons* no tienen miedo a la muerte, enviaron a un espía a explorar un sector por donde ellos podrían invadir. El espía debe encontrar la ruta más larga posible para poder llegar al planeta de los enemigos. La nave sólo se puede mover horizontal o verticalmente de un cuadrante a otro. Cada cuadrante tiene un número y este número puede repetirse en otro cuadrante. La nave debe pasar entre los cuadrantes de modo tal que no visite un cuadrante que tenga el mismo número ni visitar el mismo cuadrante. Haga un programa que determine cual es la ruta más larga dentro de un sector determinado.

Input

El archivo comenzará con una primera línea que indicará el numero de sectores que se van a evaluar. La próxima línea debe contener la cantidad de filas y columnas que contiene el sector. Luego debe seguir la tabla que tiene en cada intersección (cuadrante) un número del 0 al 99 separado por un espacio en blanco.

Output

La salida consiste en un par de números entre paréntesis que indican la coordenada inicial del cuadrante (base 1) en donde comienza la ruta. En una próxima línea se muestra la lista de números a seguir separado por un espacio en blanco. Puede haber varias soluciones que den el mismo número de pasos. En este caso se muestran todas las soluciones.

Sample Input (File: klingon.in)

```
1
3 3
 6 20  3
 3 17  9
20 10  6
```

6	20	3
3	17	9
20	10	6

Sample Output (klingon.out)

```
(1,1)
6-20-3-9-17-10
6-3-20-10-17-9

(1,2)
20-3-9-6-10-17

(1,3)
3-20-17-9-6-10
3-9-6-10-17-20

(2,1)
3-17-9-6-10-20
3-20-10-6-9-17

(2,2)
17-20-3-9-6-10
17-9-6-10-20-3
17-10-6-9-3-20
17-3-20-10-6-9

(2,3)
9-3-20-17-10-6
9-17-3-20-10-6
9-6-10-20-3-17

(3,1)
20-3-17-9-6-10
20-10-6-9-17-3

(3,2)
10-20-3-17-9-6
10-17-20-3-9-6
10-6-9-3-20-17

(3,3)
6-9-3-20-17-10
6-10-17-20-3-9
6-10-20-3-17-9
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Experto	**Universidad**: UPR – Bayamón
Autor: Juan Solá	**Tipo de competencia**: Programación
Problema #: 2	

SOLON File System Defragmentator 1.0

Definición del problema

El sistema operativo VSD-Ultrix Vista utiliza el sistema de archivos SOLON. Este sistema utiliza una tabla de direcciones de archivos (File Allocation Table) de 32 bits llamada el SOLON-FAT. Un ejemplo del SOLON-FAT sigue a continuación:

ID	Filename	FC	In-link	Start-Addr	End-Addr
1	/data/images/aoki.gif	0	0	0100	0126
2	/home/juan/ethereal.txt	1	7	20A3	210C
3	/rsync/backup/winident.dll	0	0	003B	005C
4	/etc/crontab.conf	1	0	005D	008C
5	/data/images/aoki.gif	1	1	050F	102A
6	/cyg-disk/windows/rsync.exe	1	0	0271	0460
7	/home/juan/ethereal.txt	0	0	0001	003A
8	/rsync/backup/winident.dll	1	3	0127	019F

El SOLON-FAT se compone de lo siguiente:

- Cada entrada en el SOLON-FAT está identificada con un número natural (1,2,3…) y ese es su ID dentro de la tabla. (No necesariamente comienzan en 1 pero si son sequenciales).
- Hay un campo *binario (Boolean)* que identifica el principio del archivo. Este campo se le llama el *First-Chain* (FC).
- Hay un campo llamado *in-link* que identifica la posición dentro del SOLON-FAT del próximo segmento de datos referente al archivo definido en el campo de *filename*. Si este campo está en 0 significa que este es el último segmento del archivo.
- Los campos de Start-Addr and End-Addr identifican donde comienza y termina ese segmento de datos referente al archivo descrito en *filename*. (NOTE que las posiciones son hexadecimales).

Por ejemplo el archivo `/data/images/aoki.gif` tiene 2 entradas en el SOLON-FAT (IDS 5 y 1). El archivo esta dividido en 2 segmentos. El primer segmento comienza en la posicion 050F y termina en la 102A y el segundo segmento comienza en la pos. 0100 y termina en la pos. 0126.

El programa

Usted deberá construir un programa que utilice los archivos SFAT.IN y ISO.IN y generar los archivos SFAT.OUT y ISO.OUT. El archivo SFAT.IN contiene el SOLON-FAT y el archivo ISO.IN contiene una imagen del disco. El sistema deberá utilizar el SFAT.IN para defragmentar el ISO.IN. Ejemplo:

SFAT.IN:

1	\data\a.txt		1	3	001	005
2	\data\b.txt		1	4	006	013
3	\data\a.txt		0	0	014	020
4	\data\b.txt	0	0	021	02A	

ISO.IN

CactuTerry Funk vs Jack vs. SabuRick Flair

Luego del Defrag su sistema debe generar

SFAT.OUT

1	\data\a.txt		1	0	001	014
2	\data\b.txt		1	0	015	02D

ISO.OUT
Cactus Jack vs. SabuTerry Funk vs. Rick Flair

IMPORTANTE:

No asuma que el ISO.IN es un archivo de texto. Abstraiga de que el mismo es una concatenacion de bytes sin sentido y lo que le da el sentido es el programa que los lea. Por ejemplo ISO.IN puede ser una concadenación de imágenes así como el ISO.IN del ejemplo es una concadenación de textos.

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Experto	**Universidad**: UPR – Bayamón
Autor: ACM	**Tipo de competencia**: Programación
Problema #: 3	

So Doku Checker

The best logical puzzles often are puzzles that are based on a simple idea. So Doku is one such type of puzzle. Although So Dokus have been around for some twenty years, in the last few years they conquered the world exponentially. Hundreds of newspapers and websites are now publishing them on a daily basis. For those of you unfamiliar with these puzzles, let me give a brief introduction.

		3	9			7	6	
	4				6			9
6		7		1				4
2			6	7			9	
		4	3		5	6		
	1			4	9			7
7				9		2		1
3			2				4	
	2	9			8	5		

The picture above contains an example of a Su Doku puzzle. As you can see, we have a 9X9 grid filled with single digits from 1 to 9 and empty places. The grid is further divided into nine 3X3 sub-grids, indicated by the thick lines. To solve the puzzle you have to fill the empty places with digits according to the following rules:

- Every row should contain the digits 1 to 9 exactly once;
- Every column should contain the digits 1 to 9 exactly once;
- Every 3X3 sub-grid should contain the digits 1 to 9 exactly once.

A well formed Su Doku can be solved with paper and pencil using logical deduction only. To be well formed it should be legal (no row, column or sub-grid contains a digit more than once), solvable (the empty places can all be filled while respecting the rules) and unique (there is only one solution). This is what your program is going to check.

Input

The input contains several (partially) filled grids, each representing a Su Doku puzzle. For every puzzle there are 9 lines with 9 digits giving the puzzle in row major order. Empty places in the puzzle are represented by the digit 0 (zero). Digits on a line are separated by one space. The grids are separated by one empty line.
The first grid in the sample input represents the puzzle given in the picture.

Output

For every grid in the input, determine one of the following four verdicts:

- "Illegal" if the puzzle violates one of the three rules;
- "Unique" if only one solution exists;
- "Ambiguous" if more than one solution exists;
- "Impossible" if no solution exists;
- If the problem is unique, show the solution

Print one line per grid, in the format:"Case <N>: <VERDICT>.", where N is the case number, starting from 1, and VERDICT is one of the four words in the list. See the sample output for the exact format.

Note: an "Illegal" puzzle is also "Impossible", of course, but your program should print "Illegal" in that case. Only print "Impossible" if the input doesn't violate one of the three rules, but the puzzle still can't be solved.

Sample Input

```
0 0 3 9 0 0 7 6 0
0 4 0 0 0 6 0 0 9
6 0 7 0 1 0 0 0 4
2 0 0 6 7 0 0 9 0
0 0 4 3 0 5 6 0 0
0 1 0 0 4 9 0 0 7
7 0 0 0 9 0 2 0 1
3 0 0 2 0 0 0 4 0
0 2 9 0 0 8 5 0 0

0 0 3 9 0 0 7 6 0
0 4 0 0 0 6 0 0 9
6 0 0 0 1 0 0 0 4
0 0 0 6 7 0 0 9 0
0 0 4 0 0 5 6 0 0
0 1 0 0 4 9 0 0 0
7 0 0 0 9 0 2 0 1
3 0 0 2 0 0 0 4 0
0 2 0 0 0 8 5 0 0

0 0 3 9 0 0 7 6 0
0 4 0 0 0 6 0 0 9
6 0 7 0 1 0 0 0 4
2 0 0 6 7 0 0 9 0
0 0 4 3 0 5 6 0 0
0 1 0 0 4 9 0 0 7
7 2 0 0 9 0 2 0 1
3 0 0 2 0 0 0 4 0
0 2 9 0 0 8 5 0 0

0 0 3 9 0 0 7 6 0
0 4 0 0 0 6 0 0 9
6 0 7 0 1 0 0 0 4
2 0 0 6 7 0 0 9 0
0 0 4 3 0 5 6 0 0
0 1 0 0 4 9 0 0 7
7 5 0 0 9 0 2 0 1
3 0 0 2 0 0 0 4 0
0 2 9 0 0 8 5 0 0
```

Output for Sample Input

```
Case 1: Unique.
1 5 3 9 8 4 7 6 2
8 4 2 7 3 6 1 5 9
6 9 7 5 1 2 8 3 4
2 3 8 6 7 1 4 9 5
9 7 4 3 2 5 6 1 8
5 1 6 8 4 9 3 2 7
7 6 5 4 9 3 2 8 1
3 8 1 2 5 7 9 4 6
4 2 9 1 6 8 5 7 3

Case 2: Ambiguous.
Case 3: Illegal.
Case 4: Impossible.
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Experto	**Universidad**: UPR – Bayamón
Autor: ACM	**Tipo de competencia**: Programación
Problema #: 4	

Mapping the Route

Finding a path through a maze is a popular problem for computers. In this problem, a maze will consist of a rectangular array of square cells, each of which may have walls on the north, south, east and/or west sides of the cell. One cell will be identified as the starting point, and another will be identified as the goal. Your task is to find the unique route from the starting point to the goal, label each cell in the path with its sequence in the path, identify the cells that were visited but that were not in the path, and display the maze.
The algorithm you use to find a path through the maze must be the one described below. Imagine a robot is positioned in the starting cell. The robot first attempts to go west from that cell, then north, then east, then south, in sequence. The robot can move in the selected direction if

(a) there is no wall preventing it from moving in that direction, and
(b) it has not yet been in the next cell in that direction.

When the robot reaches the goal, its trip is over. If the robot reaches a cell at which no further progress is possible, it retreats to the previous cell it occupied and attempts to move in the next untried direction.

Consider the simple maze shown on the left below. It is two cells high and three cells wide. The starting cell is labeled `S' and the goal cell is labeled `G'. When the robot starts, it would first try to move west (left), but finds a wall. It then tries to move north (up), and is again blocked by a wall. A wall also prevents it from moving east (right), so it finally tries to move south (down), and succeeds. From the new cell it will eventually move east. Here it repeats its movement algorithm. Although no wall blocks its potential westward movement, it has already 'visited' the cell in that direction, so it next tries to move north, and is successful. Unfortunately, after moving north, it finds no way to extend its path, and so it retreats to the previously occupied cell. Now it tries to move east, and is successful. From that cell it will move north, and there it finds the goal. The maze that would be displayed on the output is shown on the right below. Note that the starting cell is labeled `1', each cell in the path to the goal (including the one containing the goal) is labeled with its sequence number, and each cell that was visited but is not in the path is labeled with question marks.

```
+---+---+---+        +---+---+---+
| S |   | G |        |  1|???|  5|
+   +   +   +        +   +   +   +
|           |        |  2   3   4|
+---+---+---+        +---+---+---+
```

Input

View the maze as an array of cells, with the northernmost row being row 1, and the westernmost column being column 1. In the maze above, the starting cell is row 1, column 1, and the goal cell is row 1, column 3.

There will be one or more mazes to process in the input. For each maze there will first appear six integers. The first two give the height (number of rows) and width (numer of columns) of the maze (in

cells). The next two give the position (row and column number) of the starting cell, and the last two give the position of the goal. No maze will have more than 12 rows or 12 columns, and there will always be a path from the starting point to the goal.

Following the first six integers there will appear one integer for each cell, in row major order. The value of each integer indicates whether a cell has a wall on its eastern side (1) and whether it has a wall on its southern side (2). For example, a cell with no eastern or southern wall has a value of 0. A cell with only a southern wall has a value of 2. A cell with both an eastern and a southern wall has a value of 3. The cells on the periphery of the maze always have appropriate walls to prevent the robot from leaving the maze; these are not specified in the input data.

The last maze in the input data will be followed by six zeroes.

Output

For each maze, display the maze as shown in the example above and the expected output below, appropriately labeled and prefixed by the maze number. The mazes are numbered sequentially starting with 1.

Sample Input

```
2 3 1 1 1 3
1 1 0
0 0 0

4 3 3 2 4 3
0 3 0
0 2 0
0 3 0
0 1 0

0 0 0 0 0 0
```

Sample Output

```
Maze 1

+---+---+---+
|  1|???|  5|
+   +   +   +
|  2   3   4|
+---+---+---+


Maze 2

+---+---+---+
|??? ???|???|
+   +---+   +
|  3   4   5|
+   +---+   +
|  2   1|  6|
+   +---+   +
|       |  7|
+---+---+---+
```

Fecha: 22/abril/2006	**Nombre de la competencia**: Séptimas Competencias
Categoría: Experto	**Universidad**: UPR – Bayamón
Autor: ACM	**Tipo de competencia**: Programación
Problema #: 5	

The Boggle Game

The language PigEwu has a very simple syntax. Each word in this language has exactly 4 letters. Also each word contains exactly two vowels (y is consider a vowel in PigEwu). For instance, "maar" and "even" are legitimate words, "arts" is not a legal word.

In the game boggle, you are given a 4x4 array of letters and asked to find all words contained in it. A word in our case (PigEwu) will thus be a sequence of 4 distinct squares (letters) that form a legal word and such that each square touches (have a corner or edge in common) the next square.

For example:

```
A  S  S  D
S  B  E  Y
G  F  O  I
H  U  U  K
```

In this board a (partial) list of legal words include:

```
ASGU    SABO    FOIK    FOYD    SYDE    HUFO
```

BEBO is a legal word but it is not on this boggle board (there are no two B's here).

Write a program that reads a pair of Boggle boards and lists all PigEwu words that are common to both boards.

Input

The input file will include a few data sets. Each data set will be a pair of boards as shown in the sample input. All entries will be upper case letters. Two consecutive entries on same board will be separated by one blank. The first row in the first board will be on the same line as the first row of the second board. They will be separated by four spaces, the same will hold for the remaining 3 rows. Board pairs will be separated by a blank line. The file will be terminated by `#'.

Output

For each pair of boggle boards, output an alphabetically-sorted list of all common words, each word on a separate line; or the statement "`There are no common words for this pair of boggle boards.`"

Separate the output for each pair of boggle boards with a blank line.

Sample Input

```
D F F B    W A S U
T U G I    B R E T
O K J M    Y A P Q
K M B E    L O Y R

Z W A V    G S F U
U N C O    A H F T
Y T G I    G N A L
H G P M    B O O B

#
```

Sample Output

```
There are no common words for this pair of boggle boards.

ANGO
AOGN
GNAO
GOAN
NAOG
NGOA
OANG
OGNA
```

Universidad de Puerto Rico en Bayamón
Departamento de Ciencias Computadoras
Asociación de Estudiantes de Ciencias Computadoras

Categoría Principiante
Instrucciones generales

1. Lea detenidamente cada problema y asigne el orden en que piensan trabajarlos.
2. Los problemas se deben entregar según se van resolviendo. No espere al final.
3. Identifique el *diskette* asignado con el nombre de su pareja.
4. Asegúrese de que se ponga la fecha en la hoja de solicitud de evaluación del problema y el nombre del programa.
5. **NO** utilice la misma hoja si va a volver a someter el mismo problema de nuevo. Use otra hoja nueva.
6. Debe crear y compilar los programas en el *desktop* de su computadora. En el *diskette* sólo va a incluir el código del programa que va a someter.
7. Al final recuerde devolverle al Juez todas las hojas que se utilizaron para someter sus programas. Esto ayuda mucho al momento de determinar las posiciones.

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: Principiante	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 1	

Minesweeper

Have you ever played Minesweeper? This cute little game comes with a certain operating system whose name we can't remember. The goal of the game is to find where all the mines are located within a $M \times N$ field.

The game shows a number in a square which tells you how many mines there are adjacent to that square. Each square has at most eight adjacent squares. The 4 x 4 field on the left contains two mines, each represented by a ``*" character. If we represent the same field by the hint numbers described above, we end up with the field on the right:

```
*...
....
.*..
....
```

```
*100
2210
1*10
1110
```

Input

The input will consist of an arbitrary number of fields. The first line of each field contains two integers n and m ($0 < n, m \leq 100$) which stand for the number of lines and columns of the field, respectively. Each of the next n lines contains exactly m characters, representing the field.

Safe squares are denoted by ``." and mine squares by ``*," both without the quotes. The first field line where $n = m = 0$ represents the end of input and should not be processed.

Output

For each field, print the message `Field #x:` on a line alone, where x stands for the number of the field starting from 1. The next n lines should contain the field with the ``." characters replaced by the number of mines adjacent to that square. There must be an empty line between field outputs.

Sample Input (File: minesweeper.in)

```
4 4
*...
....
.*..
....
3 5
**...
.....
.*...
0 0
```

Sample Output (File: minesweeper.out)

```
Field #1:
*100
2210
1*10
1110

Field #2:
**100
33200
1*100
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: Principiante	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 2	

WERTYU

A common typing error is to place your hands on the keyboard one row to the right of the correct position. Then ``Q'' is typed as ``W'' and ``J'' is typed as ``K'' and so on. Your task is to decode a message typed in this manner.

Input

Input consists of several lines of text. Each line may contain digits, spaces, uppercase letters (except ``Q'', ``A'', ``Z''), or punctuation shown above [except back-quote (` )]. Keys labeled with words [`Tab`, `BackSp`, `Control`, etc.] are not represented in the input.

Output

You are to replace each letter or punctuation symbol by the one immediately to its left on the QWERTY keyboard shown above. Spaces in the input should be echoed in the output.

Sample Input (Screen)

```
O S, GOMR YPFSU
```

Sample Output (Screen)

```
I AM FINE TODAY
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: Principiante	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 3	

LC-Display

Escriba un programa que pueda mostrar utilizando caracteres, los dígitos numéricos que se indiquen. Estos dígitos deben mostrarse en formato LCD similar al que muestran las calculadoras.

Input

Se va a leer de un archivo de varias líneas. Cada línea va a tener dos números enteros separados por un espacio. El primer dato (s) indica el tamaño (size) en el que debe ser mostrado el número ($1 \leq s \leq 10$) y el segundo dato (n) indica el número que se desea mostrar ($0 \leq n \leq 99,999,999$). La última línea de datos va a tener dos ceros (0 0) indicando que no hay más números para trabajar.

Output

Muestre en pantalla los números especificados en el archivo de input en formato LCD utilizando el guión (-) para los segmentos horizontales y el (|) para los segmentos verticales. Cada dígito ocupa exactamente $s + 2$ columnas y $2s + 3$ filas. Debe haber exactamente una columna de blancos entre cada dos dígitos.

Debe haber una línea en blanco entre cada número. A continuación se muestra un ejemplo.

Sample Input

(File: LCdisplay.in)

```
2 12345
3 67890
0 0
```

Sample Output

(File: LCdisplay.out)

```
   --   --      --
  |  |  |  | |  | |
  |  |  |  | |  | |
   --   --   --   --
  | |     |  |  |  |
  | |     |  |  |  |
   --   --      --

 ---   ---   ---   ---   ---
|     | |  | |  | |  | |  |
|     | |  | |  | |  | |  |
|     | |  | |  | |  | |  |
 ---     ---   ---
|  |    | |  |    | |  | |
|  |    | |  |    | |  | |
|  |    | |  |    | |  | |
 ---     ---   ---   ---
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: Principiante	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 4	

Primary Arithmetic

Children are taught to add multi-digit numbers from right to left, one digit at a time. Many find the ``carry'' operation, where a 1 is carried from one digit position to the next, to be a significant challenge. Your job is to count the number of carry operations for each of a set of addition problems so that educators may assess their difficulty.

Input

Each line of input contains two unsigned integers less than 10 digits. The last line of input contains ```0 0`''.

Output

For each line of input except the last, compute the number of carry operations that result from adding the two numbers and print them in the format shown below.

Sample Input (File: primary.in)

```
123 456
555 555
123 594
0 0
```

Sample Output (File: primary.out)

```
No carry operation.
3 carry operations.
1 carry operation.
```

Universidad de Puerto Rico en Bayamón
Departamento de Ciencias Computadoras
Asociación de Estudiantes de Ciencias Computadoras

Categoría Intermedio
Instrucciones generales

1. Lea detenidamente cada problema y asigne el orden en que piensan trabajarlos.
2. Los problemas se deben entregar según se van resolviendo. No espere al final.
3. Identifique el *diskette* asignado con el nombre de su pareja.
4. Asegúrese de que se ponga la fecha en la hoja de solicitud de evaluación del problema y el nombre del programa.
5. **NO** utilice la misma hoja si va a volver a someter el mismo problema de nuevo. Use otra hoja nueva.
6. Debe crear y compilar los programas en el desktop de su computadora. En el diskette sólo va a incluir el código del programa que va a someter.
7. Al final recuerde devolverle al Juez todas las hojas que se utilizaron para someter sus programas. Esto ayuda mucho al momento de determinar las posiciones.

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: Intermedio	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 1	

Reverse and Add

The *reverse and add* function starts with a number, reverses its digits and adds the reverse to the original. If the sum is not a palindrome (meaning it does not give the same number read from left to right and right to left), we repeat this procedure until it does.

For example, if we start with 195 as the initial number, we get 9,339 as the resulting palindrome after the fourth addition:

195	786	1,473	5,214
591	687	3,741	4,125
+ ---	+ ---	+ ---	+ ---
786	1,473	5,214	9,339

This method leads to palindromes in a few steps for almost all of the integers. But there are interesting exceptions. 196 is the first number for which no palindrome has been found. It has never been proven, however, that no such palindrome exists.

You must write a program that takes a given number and gives the resulting palindrome (if one exists) and the number of iterations/additions it took to find it.

You may assume that all the numbers used as test data will terminate in an answer with less than 1,000 iterations (additions), and yield a palindrome that is not greater than 4,294,967,295.

Input

The first line will contain an integer N ($0 < N \leq 100$), giving the number of test cases, while the next N lines each contain a single integer P whose palindrome you are to compute.

Output

For each of the N integers, print a line giving the minimum number of iterations to find the palindrome, a single space, and then the resulting palindrome itself.

Sample Input (File: reverse.in)

```
3
195
265
750
```

Sample Output (File: reverse.out)

```
4 9339
5 45254
3 6666
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: Intermedio	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 2	

Where's Waldorf?

Given an *m* by *n* grid of letters and a list of words, find the location in the grid at which the word can be found.

A word matches a straight, uninterrupted line of letters in the grid. A word can match the letters in the grid regardless of case (i.e., upper- and lowercase letters are to be treated as the same). The matching can be done in any of the eight horizontal, vertical, or diagonal directions through the grid.

Input

The input begins with a single positive integer on a line by itself indicating the number of cases, followed by a blank line. There is also a blank line between each two consecutive cases.

Each case begins with a pair of integers *m* followed by *n* on a single line, where $1 \leq m, n \leq 50$ in decimal notation. The next *m* lines contain *n* letters each, representing the grid of letters where the words must be found. The letters in the grid may be in upper- or lowercase. Following the grid of letters, another integer *k* appears on a line by itself ($1 \leq k \leq 20$). The next *k* lines of input contain the list of words to search for, one word per line. These words may contain upper- and lowercase letters only - no spaces, hyphens, or other non-alphabetic characters.

Output

For each word in each test case, output a pair of integers representing its location in the corresponding grid. The integers must be separated by a single space. The first integer is the line in the grid where the first letter of the given word can be found (1 represents the topmost line in the grid, and *m* represents the bottommost line). The second integer is the column in the grid where the first letter of the given word can be found (1 represents the leftmost column in the grid, and *n* represents the rightmost column in the grid). If a word can be found more than once in the grid, then output the location of the uppermost occurrence of the word (i.e., the occurrence which places the first letter of the word closest to the top of the grid). If two or more words are uppermost, output the leftmost of these occurrences. All words can be found at least once in the grid.

The output of two consecutive cases must be separated by a blank line.

Sample Input (File: waldorf.in)

```
1

8 11
abcDEFGhigg
hEbkWalDork
FtyAwaldORm
FtsimrLqsrc
byoArBeDeyv
Klcbqwikomk
strEBGadhrb
yUiqlxcnBjf
4
Waldorf
Bambi
Betty
Dagbert
```

Sample Output (File: waldorf.out)

```
2 5
2 3
1 2
7 8
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: Intermedio	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 3	

Summation of Four Primes

Waring's prime number conjecture states that every odd integer is either prime or the sum of three primes. Goldbach's conjecture is that every even integer is the sum of two primes. Both problems have been open for over 200 years.

In this problem you have a slightly less demanding task. Find a way to express a given integer as the sum of exactly four primes.

Input

Each input case consists of one integer n ($n \leq 10000000$) on its own line. Input is terminated by end of file.

Output

For each input case n, print one line of output containing four prime numbers which sum up to n. If the number cannot be expressed as a summation of four prime numbers print the line ```Impossible`." in a single line. There can be multiple solutions. Any good solution will be accepted.

Sample Input (File: fourprimes.in)

```
24
36
46
```

Sample Output (File: fourprimes.out)

```
3 11 3 7
3 7 13 13
11 11 17 7
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: Intermedio	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 4	

Ant on a Chessboard

One day, an ant named Alice came upon an $M \times M$ chessboard. She wanted to explore all the cells of the board. So she began to walk along the board by peeling off a corner of the board.

Alice started at square (1, 1). First, she went up for a step, then a step to the right, and a step downward. After that, she went a step to the right, then two steps upward, and then two grids to the left. In each round, she added one new row and one new column to the corner she had explored.

For example, her first 25 steps went like this, where the numbers in each square denote on which step she visited it.

25	24	23	22	21
10	11	12	13	20
9	8	7	14	19
2	3	6	15	18
1	4	5	16	17

Her 8th step put her on square (2, 3), while her 20th step put her on square (5, 4). Your task is to decide where she was at a given time, assuming the chessboard is large enough to accept all movements.

Input

The input file will contain several lines, each with an integer N denoting the step number where $1 \leq N \leq 2 \times 10^9$. The file will terminate with a line that contains the number `0`.

Output

For each input situation, print a line with two numbers (x,y) denoting the column and the row number, respectively. There must be a single space between them.

Sample Input (File: ant.in)

```
8
20
25
0
```

Sample Output (File: ant.out)

```
2 3
5 4
1 5
```

Universidad de Puerto Rico en Bayamón
Departamento de Ciencias Computadoras
Asociación de Estudiantes de Ciencias Computadoras

Categoría Experto
Instrucciones generales

1. Lea detenidamente cada problema y asigne el orden en que piensan trabajarlos.
2. Los problemas se deben entregar según se van resolviendo. No espere al final.
3. Identifique el *diskette* asignado con el nombre de su pareja.
4. Asegúrese de que se ponga la fecha en la hoja de solicitud de evaluación del problema y el nombre del programa.
5. **NO** utilice la misma hoja si va a volver a someter el mismo problema de nuevo. Use otra hoja nueva.
6. Debe crear y compilar los programas en el desktop de su computadora. En el diskette sólo va a incluir el código del programa que va a someter.
7. Al final recuerde devolverle al Juez todas las hojas que se utilizaron para someter sus programas. Esto ayuda mucho al momento de determinar las posiciones.

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: EXPERTO	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 1	

Fmt

The UNIX program *fmt* reads lines of text, combining and breaking them so as to create an output file with lines as close to 72 characters long as possible without exceeding this limit. The rules for combining and breaking lines are as follows:

- A new line may be started anywhere there is a space in the input. When a new line is started, blanks at the end of the previous line and at the beginning of the new line are eliminated.
- A line break in the input may be eliminated in the output unless (1) it is at the end of a blank or empty line, or (2) it is followed by a space or another line break. When a line break is eliminated, it is replaced by a space.
- Spaces must be removed from the end of each output line.
- Any input word containing more than 72 characters must appear on an output line by itself.

You may assume that the input text does not contain any tabbing characters.

Sample Input (File: fmt.in)

```
   Unix fmt

The unix fmt program reads lines of text, combining
and breaking lines so as to create an
output file with lines as close to without exceeding
72 characters long as possible.  The rules for combining and breaking
lines are as follows.

   1.  A new line may be started anywhere there is a space in the input.
If a new line is started, there will be no trailing blanks at the
end of the previous line or at the beginning of the new line.

   2.  A line break in the input may be eliminated in the output, provided
it is not followed by a space or another line break.  If a line
break is eliminated, it is replaced by a space.
```

Sample Output (File: fmt.out)

```
   Unix fmt

The unix fmt program reads lines of text, combining and breaking lines
so as to create an output file with lines as close to without exceeding
72 characters long as possible.  The rules for combining and breaking
lines are as follows.

   1.  A new line may be started anywhere there is a space in the input.
If a new line is started, there will be no trailing blanks at the end of
the previous line or at the beginning of the new line.

   2.  A line break in the input may be eliminated in the output,
provided it is not followed by a space or another line break.  If a line
break is eliminated, it is replaced by a space.
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: EXPERTO	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 2	

Calculator Language

Calculator Language (CL) supports assignment, positive and negative integers and simple arithmetic. The allowable characters in a CL statement are thus:

A..Z	variable names
0..9	digits
+	addition operator
-	subtraction operator
*	multiplication operator
/	integer division operator
=	assignment operator
()	brackets
_	negative sign

All operators have the same precedence and are right associative, thus 15 - 8 - 3 = 15 - (8 - 3) = 10. As one would expect, brackets will force the expression within them to be evaluated first. Brackets may be nested arbitrarily deeply. An expression never has two operators next to each other (even if separated by a bracket), an assignment operator is always immediately preceded by a variable and the leftmost operator on a line is always an assignment. For readability, spaces may be freely inserted into an expression, except between a negative sign and a number. A negative sign will not appear before a variable. All variables are initialised to zero (0) and retain their values until changed explicitly.

Write a program that will accept and evaluate expressions written in this language. Each expression occupies one line and contains at least one assignment operator, and maybe more.

Input

Input will consist of a series of lines, each line containing a correct CL expression. No line will be longer than 100 characters. The file will be terminated by a line consisting of a single #.

Output

Output will consist of a series of lines, one for each line of the input. Each line will consist of a list of the final values of all variables whose value changes as a result of the evaluation of that expression. If more than one variable changes value, they should be listed in alphabetical order, separated by commas. If a variable changes value more than once in an expression, only the final value is output. A variable is said to change value if its value after the expression has been evaluated is different from its value before the expression was evaluated. If no variables change value, then print the message `No Change'. Follow the format shown below exactly.

Sample input *(File: calculator.in)*

```
A = B = 4
C = (D = 2)*_2
C = D = 2 * _2
F = C - D
E = D * _10
Z = 10 / 3
#
```

Sample output *(File: calculator.out)*

```
A = 4, B = 4
C = -4, D = 2
D = -4
No Change
E = 40
Z = 3
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: EXPERTO	**Universidad**: UPR- Bayamón
Autor: ACM	**Tipo de Competencia**: Programación
Problema # 3	

Check the Check

Your task is to write a program that reads a chessboard configuration and identifies whether a king is under attack (in check). A king is in check if it is on square which can be taken by the opponent on his next move.

White pieces will be represented by uppercase letters, and black pieces by lowercase letters. The white side will always be on the bottom of the board, with the black side always on the top.

For those unfamiliar with chess, here are the movements of each piece:

Pawn (`p` or `P`):
can only move straight ahead, one square at a time. However, it takes pieces diagonally, and that is what concerns you in this problem.

Knight (`n` or `N`)
: has an L-shaped movement shown below. It is the only piece that can jump over other pieces.

Bishop (`b` or `B`)
: can move any number of squares diagonally, either forward or backward.

Rook (`r` or `R`)
: can move any number of squares vertically or horizontally, either forward or backward.

Queen (`q` or `Q`)
: can move any number of squares in any direction (diagonally, horizontally, or vertically) either forward or backward.

King (`k` or `K`)
: can move one square at a time in any direction (diagonally, horizontally, or vertically) either forward or backward.

Movement examples are shown below, where ``*'' indicates the positions where the piece

can capture another piece:

```
Pawn            Rook          Bishop        Queen         King          Knight
........      ...*....      .......*      ...*...*      ........      ........
........      ...*....      *.....*.      *..*..*.      ........      ........
........      ...*....      .*...*..      .*.*.*..      ........      ..*.*...
........      ...*....      ..*.*...      ..***...      ..***...      .*...*..
...p....      ***r****      ...b....      ***q****      ..*k*...      ...n....
..*.*...      ...*....      ..*.*...      ..***...      ..***...      .*...*..
........      ...*....      .*...*..      .*.*.*..      ........      ..*.*...
........      ...*....      *.....*.      *..*..*.      ........      ........
```

Remember that the knight is the only piece that can jump over other pieces. The pawn movement will depend on its side. If it is a black pawn, it can only move one square diagonally down the board. If it is a white pawn, it can only move one square diagonally up the board. The example above is a black pawn, described by a lowercase ``p''. We use ``*move*'' to indicate the squares where the pawn can capture another piece.

Input

There will be an arbitrary number of board configurations in the input, each consisting of eight lines of eight characters each. A ``.'' denotes an empty square, while upper- and lowercase letters represent the pieces as defined above. There will be no invalid characters and no configurations where both kings are in check. You must read until you find an empty board consisting only of ``.'' characters, which should not be processed. There will be an empty line between each pair of board configurations. All boards, except for the empty one, will contain exactly one white king and one black king.

Output

For each board configuration read you must output one of the following answers:

```
Game #d: white king is in check.

Game #d: black king is in check.

Game #d: no king is in check.
```

where *d* stands for the game number starting from 1.

Sample Input (File: check.in)

```
..k.....
ppp.pppp
........
.R...B..
........
........
PPPPPPPP
K.......

rnbqk.nr
ppp..ppp
....p...
...p....
.bPP....
.....N..
PP..PPPP
RNBQKB.R

........
........
........
........
........
........
........
........
```

Sample Output (File: check.out)

```
Game #1: black king is in check.
Game #2: white king is in check.
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: EXPERTO	**Universidad**: UPR- Bayamón
Autor: Nelliud D. Torres	**Tipo de Competencia**: Programación
Problema # 4	

SQL SIMULATOR

No creo que tenga que explicar a este nivel lo que es SQL y su uso en las Bases de Datos. Vamos a crear un simulador del comando SELECT de una aplicación de Base de Datos como lo es Oracle© por ejemplo. El formato del SELECT sencillo que vamos a utilizar para crear esta simulación va a ser la siguiente:

```
SELECT atributos
   FROM entidad
   WHERE condición
      ORDER BY atributo;
```

Sólo vamos a trabajar con una tabla y las posibles combinaciones son las siguientes:

I. SELECT

SELECT *
SELECT atributo_1
SELECT atributo_1, atributo_2,…atributo_n

II. FROM

FROM entidad

III. WHERE

WHERE atributo { = | >| < | >= | <= | != } { constante | string } [{AND | OR } condition_2]

IV. ORDER BY

ORDER BY atributo [ASC | DESC];

Input

La tabla se va a cargar de un archivo secuencial y va a tener el siguiente formato:

```
nombre de la tabla

atributo_1, atributo_2,… atributo_n

tipo de dato-1, tipo de dato-2, … tipo de dato-n

valor_1, valor_2,… valor_n,

0 0
```

1. En la primera línea va a estar el nombre de la tabla sin espacios en blanco.
2. La segunda línea estará en blanco.
3. La tercera línea va a tener los nombres de los atributos.
4. La cuarta línea estará en blanco.
5. La quinta línea va a indicar el tipo de dato. X = alfanumérico(*string*), 9 = numérico entero, V = numérico real
6. La sexta línea estará en blanco.
7. De la séptima línea en adelante se encuentran los datos.
8. La última línea va a tener dos ceros (0) separados por un espacio en blanco. Esto nos va a indicar que es el final de la tabla.

Los comandos de SQL se piden por pantalla. Pueden tener más de una línea y no se pueden ejecutar hasta encontrar el punto y coma (;). Se tiene que validar las sintaxis de las instrucciones y los nombres de los atributos y tabla. El resultado se muestra interactivamente en pantalla.

Output

Se va a mostrar los resultados del SQL en pantalla. También deben salir los mensajes de errores y los encabezamientos que se van a mostrar en los ejemplos.

Sample Input (File: sql.in)

```
estudiante

nombre, numero_estudiante, edad, genero, promedio

x, x, 9, x, v

Zacarias Piedras Del Rio, 123-45-6789, 19, M, 3.34
Pepito Santo Cielo, 987-65-4321, 22, M, 2.81
Juana La Loca, 163-45-7823, 21, F, 4.00
Jose Ivan Alcara, 945-23-4562, 17, M, 1.69
Minguita Gomez, 527-83-5912, 28, F, 2.54
0 0
```

Sample Output (Screen)

```
SQL>SELECT * FROM estudiante;

       nombre             numero_estudiante  edad  genero  promedio
      --------            -----------------  ----  ------  --------
Zacarias Piedras Del Rio     123-45-6789      19     M       3.34
Pepito Santo Cielo           987-65-4321      22     M       2.81
Juana La Loca                163-45-7823      21     F       4.00
Jose Ivan Alcara             945-23-4562      17     M       1.69
Minguita Gomez               527-83-5912      28     F       2.54


SQL>SELECT nombre, edad, genero FROM estudiante;

       nombre             edad  genero
      --------            ----  ------
Zacarias Piedras Del Rio   19     M
Pepito Santo Cielo         22     M
Juana La Loca              21     F
Jose Ivan Alcara           17     M
Minguita Gomez             28     F

SQL>SELECT nombre, edd, genero
SQL>FROM estudiante;

*** Atributo invalido <edd> ***
```

```
SQL>SELECT nombre, edad, genero
SQL>FROM estudiant
SQL>;

*** Tabla invalida <estudiant> ***


SQL>SELECT nombre, edad, genero FROM estudiante
SQL>WHERE edad >= 21;

       nombre             edad  genero
      --------            ----  ------
Pepito Santo Cielo         22     M
Juana La Loca              21     F
Minguita Gomez             28     F


SQL>SELECT nombre, edad, genero FROM estudiante
SQL>WHERE edad >= 21
SQL>ORDER BY nombre ASC;

       nombre             edad  genero
      --------            ----  ------
Juana La Loca              21     F
Minguita Gomez             28     F
Pepito Santo Cielo         22     M


SQL>SELECT nombre, promedio FROM estudiante
SQL>ORDER BY promedio ASC;

       nombre             promedio
      --------            --------
Jose Ivan Alcara            1.69
Minguita Gomez              2.54
Pepito Santo Cielo          2.81
Zacarias Piedras Del Rio    3.34
Juana La Loca               4.00
```

Fecha: 7/may/2005	**Nombre de la competencia**: Sextas Competencias
Categoría: EXPERTO	**Universidad**: UPR- Bayamón
Autor: Juan Solá Sloan	**Tipo de Competencia**: Programación
Problema # 5	

Importador de datos desde COBOL al Solon Database Management System (SDBMS)

Se le ha encomendado la tarea de construir un programa que importe datos de un sistema de archivos *legacy* manejado por COBOL a un sistema de base de datos llamado SDBM----------S. El programa que importa los datos tiene como entrada un archivo en XML de la siguiente manera:

```
<?xml versión="0.01alpha" encoding="UTF-8"?>
<Schema-Format>
<Alphanumeric>30</Alphanumeric>
<Number>8</Number>
<Number>8.2</Number>
<Date>yyyymmdd</Date>
</Schema-Format>
</xml>
```

SDBMS solo maneja 3 tipos de datos: (1) Alfanumérico, (2) Números y (3) Fechas. Note que entremedio de las etiquetas (*tags*) de XML se encuentra los *descriptores* de los tipos de datos.

Alfanumérico solo puede llevar un número entero entre ambas etiquetas y este número indica la cantidad de espacios que SDBMS espera guardar en la base de datos (Favor ver el ejemplo). Si el archivo que se importa de COBOL tiene más que el número indicado estos caracteres no se leen, es decir, solo se guardan los caracteres indicados en la base de datos. El sistema debe reportar la cantidad de caracteres que no incluyo en la importación.

La etiqueta para el número solo lleva entremedio un valor que indica la cantidad de espacios reservados para guardar un valor numérico. En el ejemplo <Number>8</Number> significa un entero de un máximo de 8 espacios. <Number>8.2</Number> significa un número real con un máximo de 8 espacios y 2 decimales. **De encontrar un valor no numérico donde se supone que exista uno el sistema emitirá un error!** El espacio máximo que puede haber reservado son 12 espacios para los enteros y 6 para los decimales. De haberse sobrepasado de estos limites el sistema indicara el siguiente mensaje: "Valor numérico fuera de formato".

Las fechas se leen **SIEMPRE** con el año primero seguido por el número del mes y finalmente con el numero de día. Por ejemplo: 20050411 es la fecha 04/11/2005. El sistema debe validar que la entrada de la fecha sea correcta. SDBMS funciona con una ventana de fechas con años desde 1985 – 2084. Los valores para los meses son enteros positivos donde $m < 13$. Los días son enteros positivos donde $n < 32$ excepto para el mes de febrero. **No se preocupe por validar años bisiestos, PERO recuerde que hay meses de 30, 28 o 31 días.** De haber error en el formato de entrada para la fecha favor de indicar **en donde** existe el error de entrada. **Ejemplo de un mensaje**: "En la fecha 02/33/2005 existe un error en el día del mes".

OJO: FAVOR DE NOMBRAR EL ARCHIVO CON ESTE NOMBRE: SDBMS-IN.TXT

Sample Input (File: SDBMS-IN.TXT)

Ejemplo de entrada de un archivo:

```
<?xml versión="0.01alpha" encoding="UTF-8"?>
<Schema Format>
<Alphanumeric>10</Alphanumeric>
<Number>5.2</Number>
<Date>yyyymmdd</Date>
<Alphanumeric>1</Alphanumeric>
</Schema Format>
</xml>
7   <- Cantidad de records
Manchego,12345.67,19950116,Z
El hombre de cemento,76543.21,19880123,9
Mordor,M15.32,20051201,F
MC80487,0.51,19880207,A
Trinity,6.67,21050201,A
Saruman,123*.0K,19883301,8
Gandalf,200.00,19921103,The White Wizard
```

Sample Output (File: SDBMS-OUT.TXT)

El sistema debe imprimir una lista debe leer el archivo de entrada y luego generar el siguiente *output*:

```
Registro 1: Importado sin errores.
Registro 2: Importado sin incluir 10 caracteres alfanuméricos para
el campo 1.
Registro 3: Error: Carácter no permitido en campo numérico
Registro 4: Importado sin errores
Registro 5: Error: En la fecha 02/01/2105 existe un error en el año
Registro 6: Error: Carácter no permitido en campo numérico. En la
fecha 33/01/1988 existe un error en el valor del mes.
Registro 7: Importado sin incluir 15 caracteres del valor
alfanumérico para el campo 4.
```

Universidad de Puerto Rico
Bayamón, Puerto Rico

Quintas Competencias de Programación
2004

Experto

Fecha: 24/abril/2004
Categoría: Experto
Autor: 3a COMPETENCIA IBEROAMERICANA DE INFORMÁTICA
Problema #: 1

Nombre de la competencia: Quintas Competencias
Universidad: UPR – Bayamón
Tipo de competencia: Programación

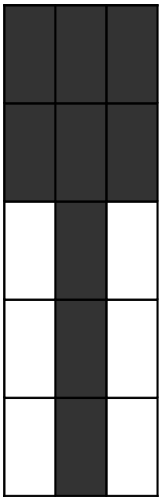
ENCAJAR

Input File Name: ENCAJAR.IN
Output File Name: <pantalla>

Narrativa

En un taller el orden es muy importante ya que es indispensable tener organizadas todas las herramientas para facilitar su ubicación. Pensando en esto se ha colgado un tablero en la pared sobre el cual se colocan cada una de las herramientas. Aunque es una buena solución, se hace problemático agregar nuevas herramientas en el tablero cuando ya no queda mucho espacio libre.

En busca de una solución a este problema se han escaneado en blanco y negro el tablero y la herramienta que se desea colgar. El escaneo se compone de un arreglo de zonas cuadradas en las cuales una "X" indica una zona cubierta por una herramienta y un punto (.) una zona libre. Por ejemplo, la forma y escaneo de un martillo son:



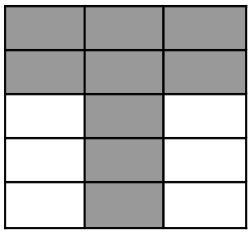
```
   X
XX
   X
XX
   .
X.
   .
X.
   .
X.
```

La nueva herramienta será maciza, lo que significa que todos sus puntos son adyacentes entre sí (compartiendo al menos un vértice). Además en su escaneo se han eliminado filas y columnas vacías.

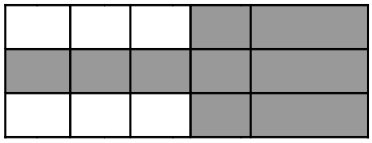
Problema

Se desea que usted elabore un programa que solucione automáticamente este problema teniendo como entrada los escaneos del tablero de herramientas y de la nueva herramienta que se quiere colgar. Debe tomar en cuenta que la nueva herramienta sólo puede ser rotada en ángulos múltiplos de 90º, por lo que sólo habrán cuatro (4) formas posibles de ubicar la herramienta dentro del tablero, designadas con los códigos [0..3]:

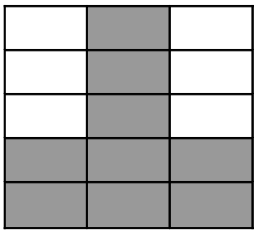
0 PARA 0°



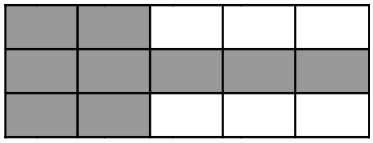
1 PARA 90º



2 PARA 180°



3 PARA 270°



La nueva herramienta debe ocupar una zona del tablero tal que no utilice zonas ya ocupadas por otras herramientas (no podría colgarse).

Entrada: ENCAJAR.IN

Este archivo contiene información que representa el escaneo del tablero y de la herramienta.

Línea 1: f_t c_t ($1 <= f_t, c_t <= 200$). Número de filas y columnas del escaneo del tablero.

Línea: 2..f_t+1: Una cadena de ct caracteres ['X','.'] representando una línea del escaneo.

Línea: f_t+2: f_h, c_h ($1 <= f_h, c_h <= 100$). Número de filas y columnas del escaneo de la herramienta.

Líneas: f_t+3..f_t+f_h+2: Una cadena de ch caracteres ['X','.'] representando una línea del escaneo.

Salida: ENCAJAR.OUT

Línea 1: *f, c*. Las coordenadas de la posición de la esquina superior izquierda de la herramienta dentro del tablero. Este punto coincide con la posición (1, 1) del escaneo de la herramienta.

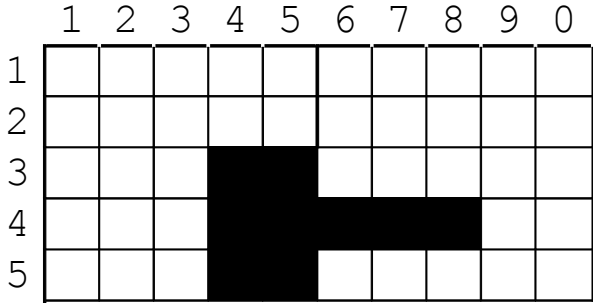
Línea 2: El código de la forma en que se colocará la figura, de acuerdo a lo descrito anterior-mente. Es posible más de una salida válida para una entrada, usted debe reportar sólo una de ellas. Si no es posible ubicar la nueva herramienta, el archivo de salida debe contener el mensaje "No hay solución".

Línea 3 en adelante: El dibujo del tablero incluyendo la herramienta nueva. Se utilizará el asterisco para indicar en donde está la nueva herramienta.

<u>Ejemplo</u>:

ENCAJAR.ENT	ENCAJAR.SAL
5	3
10	4
XXX…	3
..X.	
X…	
XXX.X.	
XXX…	XXX…
..X.	..X.
X……	X…
.X.	XXX.X.
X…	XXX**
..X…	…X.
5	X..**
3	***X.
XXX	X..**
	.X…
XXX	
.X.	
.X.	
.X.	

DIAGRAMA VISUAL DE LA NUEVA HERRAMIENTA EN EL TABLERO



Fecha: 24/abril/2004
Catgoría: Experto
Autor: ACM International Collegiate
Problema #: 2

Nombre de la competencia: Quintas competencias
Universidad: UPR – Bayamón
Tipo de competencia: Programación

Morse Mismatches

Input File Name: morse.in

Samuel F. B. Morse is best known for the coding scheme that carries his name. Morse code is still used in international radio communication. The coding of text using Morse code is straightforward. Each character (case is insignificant) is translated to a predefined sequence of *dits* and *dahs* (the elements of Morse code). Dits are represented as periods (".") and dahs are represented as hyphens or minus signs ("-"). Each element is transmitted by sending a signal for some period of time. A dit is rather short, and a dah is, in perfectly formed code, three times as long as a dit. A short silent space appears between elements, with a longer space between characters. A still longer space separated words. This dependence in the spacing and timing of elements means that Morse code operators sometimes do not send perfect code. This result in difficulties for the receiving operator, but frequently the message can be decoded depending on context.

In this problem we consider reception of words in Morse code without spacing between letters. Without the spacing, it is possible for multiple words to be coded the same. For example, if the message "dit dit dit" were received, it could be interpreted as "EEE, EI, IE" or "S" based on the coding scheme shown in the sample input. To decide between these multiple interpretations, we assume a particular context by expecting each received word to appear in a dictionary.

For this problem your program will read a table giving the encoding of letters and digits into Morse code, a list of expected words (*context*), and a sequence of words encoded in Morse code (*morse).* These *morse* words may be flawed. For each *morse* word, your program is to determine the matching word from *context*, if any. If multiple words from *context* match *morse*, or if no word matches perfectly, your program will display the best matching word and a mismatch indicator.

If a single word from *context* matches *morse* perfectly, it will be displayed on a single line, by itself. If multiple *context* words match *morse* perfectly, then select the matching word with the fewest characters. If this still results in an ambiguous match, any of these matches may be displayed. If multiple *context* words exist for a given *morse*, the matching word will be displayed followed an exclamation point ("!").

We assume only a simple case of errors in transmission in which elements may be either truncated from the end of a *morse* word or added to the end of a *morse* word. When no perfect matches for *morse* are found, display the word from *context* that matches the longest prefix of *morse*, or has the fewest extra elements beyond those in *morse*. If multiple words in *context* match using these rules, any of these matches may be displayed. Words that do not match perfectly are displayed with a question mark ("?") suffixed.

The input data will only contain cases that fall within the preceding rules.

Input

The Morse code table will appear first and consists of lines each containing an uppercase letter or a digit C, zero or more blanks, and a sequence of no more than six periods and hyphens giving the Morse code for C. Blanks may precede or follow the items on the line. A line containing a single asterisk ("*"), possibly preceded or followed by blanks, terminates the Morse code table. You may assume that there will be Morse code given for every character that appears in the *context* section.

The *context* section next, with one word per line, possibly preceded and followed by blanks. Each word in context will contain no more than ten characters. No characters other than upper case letters and digits will appear. Thered will be at most 100 context words. A line containing only a single asterisk ("*"), possibly preceded or followed by blanks, terminates the context section.

The remainder of the input contains *morse* words separated by blanks or end-of-line characters. A line containing only a single asterisk ("*"), possibly preceded or followed by blanks, terminates the input. No *morse* word will have more than eighty (80) elements.

Output
For each input *morse* word, display the appropriate matching word from *context* followed by an exclamation mark ("!") or question mark ("?") if appropriate. Each word is to appear on a separate line starting in column one.

Sample Input

```
A    .-
B    -…
C    -.-.
D    -..
E    .
F    ..-.
G    --.
H    ….
I    ..
J    .---
K    -.-
L    .-..
M    --
N    -.
O    ---
P    .--.
Q    --.-
R    .-.
```

```
*
AN
EARTHQUAKE
EAT
GOD
HATH
IM
READY
TO
WHAT
WROTH
*
.--…..--      …..--….
 --.----..      .--.-.----..
.--…..--      .--.
..-.-.-….--.-..-.--.-.
..--      .-…--..-.--
----              ..--
```

```
S    …
T    -
U    ..-
V    ..-
X    -..-
Y    -.--
Z    --..
0    ------
1    .-----
2    ..---
3    …--
4    ….-
5    ….-
6    -….
7    --…
8    ---..
9    ----.
```

```
*
```

Output for the Simple Input

```
WHAT
HATH
GOD
WROTH?
WHAT
     AN
     EARTHQUAKE
     IM!
READY
TO
IM!
```

Fecha: 24/abril/2004
Categoría: Experto
Autor: Juan M. Solá
Problema #: 3

Nombre de la competencia: Quintas Competencias
Universidad: UPR- Bayamón
Tipo de Competencia: Programación

L-FAT DEFRAG. EXE

Input File Name: L-FAT. TXT
Output File Name: DEFRAG. TXT

L- FAT es la tabla de localización de archivos utilizada por LECHON-DOS. Usted debe construir un programa que dado un L-FAT fragmentado, deba producir un L-FAT defragmentado.

Ejemplo:
L-FAT fragmentado:

Start at	ID	Filename	Start at	End at	Next ID
1	C01	monstro. mpg	0800	0875	A02
1	M0450	File1. dll	108964	109991	C2013
0	F901	monstro. mpg	1519	3751	*
1	U0561	Xfig.rpm	3752	5000	*
0	J012	File1.dll	7901001	8502658	*
0	A02	Mostro.m pg	1001	1352	F901
0	C2013	File1.dll	0876	1000	J012

L-FAT defragmentado:

Start at	ID	File Name	Start at	End at	Next ID
1	M0450	File1.dll	0	602808	*
1	C01	mostro.m pg	602809	605467	*
1	U0561	Xfig.rpm	605468	606716	*

Note lo siguiente:

1. L-FAT defragmentado se organiza por orden alfabético en la columna de “filename”.
2. Los archivos se colocan empezando en la posición 0 en el disco duro.
3. El comienzo de un archivo está marcado con un 1 en la primera columna.
4. Se preserva el ID del principio del archivo al defragmentar el L-FAT.
5. Un 0 en la primera columna identifica a un fragmento en el L-FAT fragmentado.

6. Los brincos hacia otros fragmentos en el L-FAT fragmentado pueden ser hacia un ID más abajo o más arriba en la lista.
7. Al defragmentar un archivo puede ocurrir que otro tenga que ser movido del L-FAT para qaue este primer archivo no le pase por encima al segundo archivo.

Ejemplo:

<input> **L-FAT.TXT**

Start at	ID	Filename	Start at	End at	Next ID
1	C01	monstro.mpg	0800	0875	A02
1	M0450	File1. dll	108964	109991	C2013
0	F901	monstro.mpg	1519	3751	*
1	U0561	Xfig.rpm	3752	5000	*
0	J012	File1.dll	7901001	8502658	*
0	A02	Mostro.mpg	1001	1352	F901
0	C2013	File1.dll	0876	1000	J012

<output> **DEFRAG.TXT**

Start at	ID	File Name	Start at	End at	Next ID
1	M0450	File1.dll	0	602808	*
1	C01	mostro.mpg	602809	605467	*
1	U0561	Xfig.rpm	605468	606716	*

Fecha:24/abril/2004
Categoría: Experto
Autor: Juan M. Solá
Problema #:4

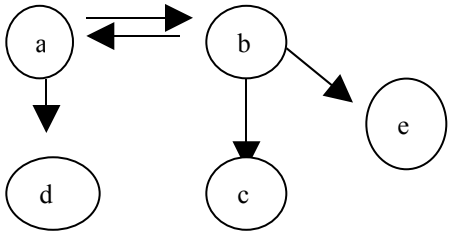
Nombre de la competencia: Quintas Competencias
Universidad: UPR- Bayamón
Tipo de competencia: Programación

Diversión Con Grafos

Input File Name: Grafo.txt
Output File Name: <pantalla>

Un grafo dirigido se representa con G=(V,E) donde V es el conjunto de los nodos y E es el conjunto de las artistas. Un ejemplo de su archivo de entrada es:

5 <Numero de Nodos>
a
b
c
d
e
4 <Numero de artistas>
a,b
b,a
d,a
b,c



G=(V,E)

V={a,b,c,d,e} y E={(a,b), (d,a), (b,c)}

Debe clasificar los nodos de la siguiente forma

Nodo a es fuertemente conexo con b
Nodo b es fuertemente conexo con a
Nodo b es conexo con c
Nodo d es conexo con a
Nodo e es disconexo

Nota

1. No existen ciclos en los grafos dados de entrada. Solo puede clasificar en tres clases: *Fuertemente conexo, conexo o disconexo.*
2. Un nodo a es fuertemente conexo con b, si existe un arista del nodo a al nodo b y del nodo b al nodo a (Ejemplo: (a,b) y (b,a)).
3. Un nodo es conexo si existe una arista de salida a otro nodo (Ejemplo: (d,a))
4. Un nodo es disconexo si no existe una arista de salida ni de entrada hacia el. Ejemplo: (nodo e).
5. Un nodo que es fuertemente conexo, también es un nodo conexo. Por ende NO debe listar al nodo a como conexo con b ya que es fuertemente conexo con el nodo b.

Orejita: Utilizar matriz de adyacencia

Universidad de Puerto Rico
Bayamón, Puerto Rico

Quintas Competencias de Programación
2004

Intermedio

Fecha: 24/abril/2004 **Nombre de la competencia**: Quintas Competencias
Categoría: Intermedio **Universidad**: UPR- Bayamón
Autor: 4to Competencia Iberoamericana de Informática **Tipo de competencia**: Programación
Problema #: 1

PARENT

Input File Name: PARENT.IN
Output File Name: PARENT. OUT

Descripción

Se tiene una secuencia de tamaño par de paréntesis en dos estados posibles abiertos '(' y cerrados ')' y se desea cambiar algunos de forma tal que se obtenga una secuencia válida de paréntesis abiertos y cerrados.

Tarea
Dada una secuencia de tamaño par de hasta 1.000 paréntesis, encontrar el número de paréntesis que deben cambiar su estado para que sea una secuencia válida.

Ejemplo:

Sea la secuencia))((((

La secuencia no es válida, para hacerla válida se pueden hacer en dos pasos, una de sus soluciones es la siguiente: cambiando las posiciones 1,3 y 6 se obtiene ()()(), siendo esta una secuencia de paréntesis válida.

Entrada: **PARENT.IN**

En la primera y única línea del archivo aparece una cadena de longitud par L (O <=L <= 1.000) formada por los caracteres '(' y ')' sin espacios entre ellos.

Salida: **PARENT.OUT**

En la primera línea del archivo de salida debe aparecer el valor de M, indicando el mínimo de paréntesis que se deben cambiar para obtener una secuencia válida y en la segunda línea M enteros separados por espacio indicado las posiciones donde se deben hacer los cambios. Si no es necesario hacer cambios, el archivo de salida sólo contendrá una línea con el número 0.

EJEMPLO

PARENT. IN	**PARENT. OUT**
))((((	3 146

Fecha: 24/abril/2004 **Nombre de la competencia**: Quintas Compencias
Categoría: Intermedio **Universidad**: UPR- Bayamón
Autor: 1992 ACM Scholastic Programming Contest **Tipo de Competencia**: Programación
Problema: #2

Spreadsheet Calculator

Input **File Name:** SPRCAL.IN
Output **File Name:** SPRCAL. OUT

A spreadsheet is a rectangular array of cells. Cells contain data or expressions that can be evaluate to obtain data. A "simple" spreadsheet is one in which data are integers and expressions are mixed sums and differences of inters and cell references. For any expression, if each cell that is referenced contains an integer, then the expression can be replaced by the integer to which the expression evaluates. You are to write a program which evaluates simple spreadsheets.

Input

Input consists of a sequence of simple spreadsheets. Each spreadsheet begins with a line specifying the number of rows and the number of columns. No spreadsheet contains more than 20 rows or 10 columns. Rows are labeled by capital letters A through T. Columns are labeled by decimal digits 0 through 9. Therefore, the cell in the first row and first column is referenced as A0; the cell in the twentieth row and fifth columns is referenced as T4.

Following the specification of the number of rows and columns is one line of data for each cell, presented in row-major order.)That is, all cells for the first row come first, followed by all cells for the second row, etc.) Each cell initially contains a signed integer value or an expression involving unsigned integer constants, cell references, and the operators + (addition) and – (subtraction). IF a cell initially contains a signed integer, the corresponding input line will begin with an optical minus sign followed by one or more decimal digits. If a cell initially contains an expression, its input line will contain one or more cell references or unsigned integer constants separated from each other by + and – signs. Such a line must begin with a cell reference. No expression contains more than 75 characters. No line of input contains leading blanks. No expression contains any embedded blanks. However, any line may contain trailing blanks.

The end of the sequence of spreadsheet is marked by a line specifying 0 rows and 0 colums.

Output

For each spreadsheet in the input, you are to determine the value of each expression and display the resulting spreadsheet as a rectangular array of number with the rows and columns appropriately labeled. In each display, all numbers for a column must appear tight-justified and aligned with the column label. Operators are evaluated left to right in each expression; values in cells are always less than 10000 in absolute value. Since expressions may reference cells that themselves contain expressions, the order in which cells are evaluated is dependent on the expressions themselves.

If one more cells in a spreadsheet contain expressions with circular references, then the output for that spreadsheet should contain only a list of the unevaluated cells in row-major order, one per line, with each line containing the cell label, a colon, a blank, and the cell's original expression.

A blank line should appear following the output for each spreadsheet. Sample input and output are below.

Sample Input

```
2   2
Al+B1
5
3
BO-Al
2   2
AO
5
Cl
7
Al+Bl
BO+Al

0     0
```

Output for the Sample Input

```
      0     1
A       3     5
      B       3    -2

AO:   AO
      BO:   Cl
      Cl:
```

Fecha: 24/abril/2004
Categoría: Intermedio
Autor: Nelliud D. Torres
Problema #: 3

Nombre de la competencia: Quintas Competencias
Universidad: UPR- Bayamón
Tipo de competencia: Programación

ESCALERA ARITMÉTICA

Los antiguos Griegos descubrieron como construir el largo de los números irracionales utilizando el teorema de Pitágoras. Ellos utilizaban polígonos y conceptos de límites para calcular el área de un círculo. También desarrollaron un tipo de escalera aritmética utilizando radios para aproximar el valor de los números irracionales. Trabaja de la siguiente forma: Se forman dos columnas y las dos van a comenzar con el número uno(1). Si lo llevamos a cinco pasos, la escalera sería la siguiente:

PASOS	LADDER #1	LADDER #2
0	1	1
1	2	3
2	5	7
3	12	17
4	29	41
5	70	99

A partir del paso uno, los números de la columna llamada "LADDER #1" se forman utilizando el siguiente procedimiento:

Paso 1: 1 +1 = 2
Paso 2: 2 + 3 = 5
Paso 3: 5 + 7 = 12
Paso 4: 12 + 17 = 29
Paso 5: 29 + 41 = 70

Los números de la columna llamada "LADDER #2" se forman utilizando el siguiente procedimiento:

Paso 1: 1 + 2 = 3
Paso 2: 2 + 5 = 7
Paso 3: 5 + 12 = 17
Paso 4: 12 + 29 = 41
Paso 5: 29 + 70 = 99

Desarrolle un programa que le pida al usuario dos números iniciales y la cantidad de pasos que desea ver (de uno(1) a un máximo de veinte(20)).

Ejemplo Corrida:

Entre los dos números iniciales (1 – 99): **3 5**

Entre la cantidad de pasos que desea cotejar (1 -20): **22**
Número incorrecto, trate de nuevo
Entre la cantidad de pasos que desea cotejar (1 – 20): **6**

PASOS	LADDER #1	LADDER #2
0	3	5
1	8	11
2	19	27
3	46	65
4	111	157
5	268	379
6	647	915

Fecha: 24/abril/2004
Categoría: Intermedio
Autor: Nelliud D. Torres
Problema #: 4

Nombre de la competencia: Quintas Competencias
Universidad: UPR – Bayamón
Tipo de competencia: Programación

NUMBER PROPERTIES

A positive integer is considered prime if it is evenly divisible only by itself and 1. Also, by convention, 1 is not itself a prime. Thus, the sequence of primes begins:
2,3,5,7,11,13,17,…

A positive integer is called perfect if it is the sum of its proper divisors. For example, 28 is a perfect number because 28 = 1+2+4+7+14.

Your assignment is to write a program that will input a sequence of integers, one per line, and output the properties of each integer in the following format:

If the number is neither prime nor perfect, output "Dull".
If the number is prime but not perfect, output "Prime".
If the number is perfect but not prime, output : "Perfect".
If the number is both prime and perfect, output "This program doesn't work".

All inputs will be positive integers. The end of input will be market by the number 0 (no output should be generated for 0).

Example:

```
Input

28
7
140
1
5
0

Output

Perfect
Prime
Dull
Dull
                                        Prime
```

Universidad de Puerto Rico
Bayamón, Puerto Rico

Cuartas Competencias de Programación 2003

Experto

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Experto	**Universidad**: UPR-Bayamón
Autor: Furman University 2000	**Tipo de competencia**: Programación
Problema #: 1	

A Real Puzzler

Input File Name: **prob1.in**
Output: **to the screen**

Description:

A children's puzzle that was popular in the days before video games consisted of a 5 by 5 frame which contains 24 small tiles of equal size. A unique letter of the alphabet was printed on each small tile. Since there were only 24 tiles within the frame, the frame also contained an empty position which was the same size as a small tile. A tile could be moved into that empty position if it were immediately to the right, to the left, above or below the empty position. The object of the puzzle was to slide tiles into the empty position so that the frame displayed the letters in alphabetical order. The illustration below represents a puzzle in its original configuration and in its configuration after the following sequence of 6 moves:

1. The tile above the empty position is moved.
2. The tile to the right of the empty position is moved.
3. The tile to the right of the empty position is moved.
4. The tile below the empty position is moved.
5. The tile below the empty position is moved.
6. The tile to the left of the empty position is moved.

T	**R**	**G**	**S**	**J**
X	**D**	**O**	**K**	**I**
M		**V**	**L**	**N**
W	**P**	**A**	**B**	**E**
U	**Q**	**H**	**C**	**F**

T	**R**	**G**	**S**	**J**
X	**O**	**K**	**L**	**I**
M	**D**	**V**	**B**	**N**
W	**P**		**A**	**E**
U	**Q**	**H**	**C**	**F**

Input for your program consists of several puzzles. Each is described by its initial configuration and a sequence of moves on the puzzle. The first 5 lines of each puzzle description are the starting configuration. The next line gives the sequence of moves.

The first line of the input corresponds to the top line of tiles in the puzzle. The other lines follow in order. The empty position is indicated by a blank. Each input line contains exactly 5 characters, beginning with the character on the leftmost tile (or a blank if the leftmost tile is the empty frame position).

The sequence of moves is represented by a one line sequence of As, Bs, Rs and Ls to denote which tile moves into the empty position. A denotes that the tile above the empty position moves; B denotes that the tile below the empty position moves; L denotes that the tile to the left of the empty position moves; R denotes that the tile to the right of the empty position moves.

Input will be terminated with 1 zero (0) in the puzzle description.

Output for each puzzle begins with a puzzle label (Puzzle #1, Puzzle #2, etc.). The final configuration of the puzzle should then be printed. Format each line for a final configuration so that there is a blank character between two adjacent letters. Treat the empty tile the same as a letter. Separate output from different puzzle records by one blank line.

Example:

Input:

```
TRGSJ
XDOKI
M VLN
WPABE
UQHCF
ARRBBL
ABCDE
FGHIJ
KLMNO
PQR S
TUVWX
RAAALLLL
0
```

Output:

```
Puzzle #1:
T R G S J
X O K L I
M D V B N
W P   A E
U Q H C F

Puzzle #2:
  A B C D
F G H I E
K L M N J
P Q R S O
T U V W X
```

Additional notes:

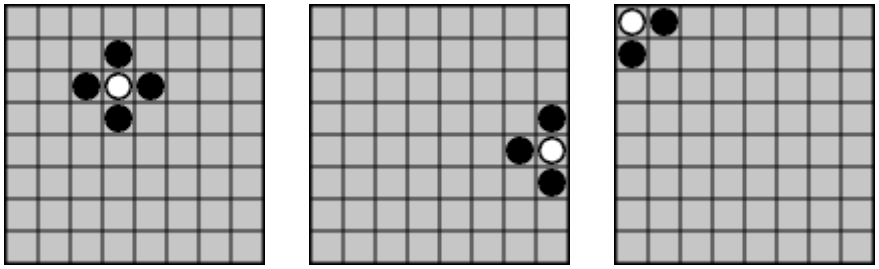
You may assume that the input will correspond to a legitimate puzzle. In addition, all sequences of moves will be valid and will not move outside the bounds of the puzzle. No sequence will be longer than 80 characters.

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Experto	**Universidad**: UPR-Bayamón
Autor: ACSL-1998	**Tipo de competencia**: Programación
Problema #: 2	

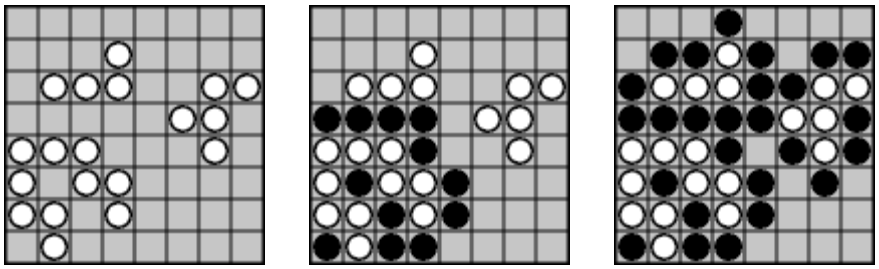
Go

The game of Go is played on an 8-by-8 board. There are two players, white (W) and black (B). The players take turns placing a piece of their color on the board, with the goal of capturing the other player's piece. You capture your opponent's piece by surrounding it with your own pieces on the horizontal and vertical sides.

The boards below illustrate the capture of a single white piece, when the white piece is in the interior, edge, and corner (from left-to-right). Note that only two of your pieces are needed to capture an opponent's piece at a corner, and three pieces are needed when the opponent is on a square along an edge.



To make this program more challenging (after all, this is the All-Star Contest), you need to check for "globs" of pieces. That is, groups of pieces of one color, all touching each other horizontally and/or vertically, that are completely surrounded horizontally and vertically by pieces of the other color. The board at the left illustrates 3 white globs; in the middle board, the glob at the lower-left has been captured using the fewest number of black pieces; and in the right board, all three globs have been captured, again using the fewest number of black pieces possible:



The input to this program will be 5 sets of data. Each set will consist of a board configuration. (We'll tell you the format later.) You need to report the best move for each player. By best move, we mean the place on the board that would capture the most opponent's pieces. If there are no such locations, print "none." If there is more than one location, you must print all. The best move for white for each configuration is worth 1 point each; the best move for black for each board configuration is worth 1 point each.

To make typing the data easy on the proctors, we'll encode the 64 locations on the board in binary for each player. That is, a 64-bit string will give all positions where black pieces reside, and a different 64-bit string will give the white pieces. (ANDing the two strings together is guaranteed to result in a bit-string that is all zero's. Think about it.) The board is encoded from left-to-right, bottom-to-top. That is, the lower left corner is the leading bit of the 64-bit string, the cell above it the 9th most significant bit, and so on.

The bit-strings will be converted to base 16. The rightmost board in the second set of images above is encoded as B028 4A15 F98C 6B10 for black and as 40D0 B0E2 0673 1000 for white.

As mentioned, for each board configuration, print the best move each player could make if he were to go next. Each set of input results in two outputs, each worth one point. Print the location of the best move(s) as a number from 1 though 64, mapping to the position of that cell in the bit string. For instance, the white piece in the leftmost board in the top set of images is at cell 44. If no move would result in the capture of any of the opponent's players, print the string "no win."

Sample Input:

```
Line 1: 0004 1801    0530    5209    0001     4410   42C8    2980
Line 2: 1204 0008    1200    2104    0180    2280    0084    1080
```

Sample Output:

```
Output 1: black: 48 and 59
Output 2: white: 63
Output 3: black: 16
Output 4: white: no win
```

Programming Problem #2 – Go - Experto

Test Data Input:

```
 #1: 00C8 1100 4240 8881 8104 0840 2520 4202
 #2: 4020 0C30 4A40 3090 3142 8200 3030 0501
 #3: FF00 E700 9918 E700 00E7 1899 00E7 00FF
 #4: 8020 0402 0090 2110 0002 3840 0168 0000
 #5: 8810 2002 8942 10B8 03A2 8008 5288 8401
```

Test Data Output:

```
 #1: black: 2
 #2: white: 56
 #3: black: 45
 #4: white: 1
 #5: black: 12 and 13
 #6: white: 52 and 53
 #7: black: no win
 #8: white: no win
 #9: black: 50
#10: white: 2, 25, 38, and 58
```

Note: There are 5 inputs; each input results in two outputs, one for black and one for white. The labels "black" and "white" are optional. Outputs 5, 6, and 10 have more than one value; all must be present to receive credit, and they may appear in any order. Outputs 7 and 8 must be the string "no win."

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Experto	**Universidad**: UPR-Bayamón
Autor: ACSL-1998	**Tipo de competencia**: Programación
Problema #: 3	

3D Tic-Tac-Toe

This problem takes the game of tic-tac-toe to a higher dimension. Consider three tic-tac-toe boards aligned above each other such that there are 27 cubes in which to place a marker. The game is played between two players, X and O, by taking turns placing a marker in one of the cubes. The winner is the first player to get three in a row. Of course, the cubes do not need to be from the same board; however, connecting the centers of the cubes does need to form a straight line.

There will be 10 sets of data. Each set consists of an initial configuration of X's and O's, followed by the placement of the next "X".

The initial configuration is given by a string that is 27 characters long consisting of X's, O's, and *'s (for blanks). The string `O*O**X*XOO*X*X*OO*X*X*X***O` represents the following three boards:

O	*	O
*	*	X
*	X	O

Bottom

O	*	X
*	X	*
O	O	*

Middle

X	*	X
*	X	*
*	*	O

Top

The placement of the next "X" is given by an integer, 1 through 27, representing the 27 cubes in the board. The numbering scheme is the same order as the 27-character long string representing the initial configuration. For example, the O's on the boards above are at positions 1, 3, 9, 10, 16, 17, and 27.

For each input set, determine if "X" wins, and if so, which two other cubes on the boards contribute to the win. If "X" wins in more than one way, you must print all ways. If "X" cannot win, print a message to that effect.

Sample Input:

```
Line 1: O*O**X*XOO*X*X*OO*X*X*X***O, 5
Line 2: XO**X*O**O***X*XXO**X***O*O, 11
```

Sample Output:

```
Output 1: 14 and 23
Output 2: 14 and 17; 1 and 21
```

Programming Problem #3 - **3D Tic-Tac-Toe -Experto**

Test Data Input:

```
 #1: *XO**XO*OOO**X*O*X**O*X*X*X, 22
 #2: *X*OX*XOO*OXX****O*O*X**O*X,  3
 #3: X*X*X*O*OO*O*O*X*XX*X***O*O, 23
 #4: XO**X*OX*O*OXOOX*X*O**XOOXX, 17
 #5: *OOXO*XX*OX*OO*XOXOOX*X**X*, 25
 #6: OO*O*XXXO*OOXOOX*X*X*XX*O**, 19
 #7: OX**OOO*X*XXOO******XX*****, 15
 #8: ***OX*XOOX*OX*XOOX*OX*X***O, 14
 #9: O**XX*OOX*O***O**XOXX*XX*OO, 14
#10: *O**OXOXOOXOXXOX*XXOXO*XOXO, 23
```

Test Data Output:

```
 #1: 6 and 14
 #2: 5 and 7
 #3: NO WIN
 #4: 16 and 18; 8 and 26
 #5: 7 and 16; 21 and 23
 #6: 7 and 13
 #7: 9 and 21
 #8: 5 and 23; 7 and 21; 10 and 18; 13 and 15
 #9: 5 and 23; 4 and 24
#10: NO WIN
```

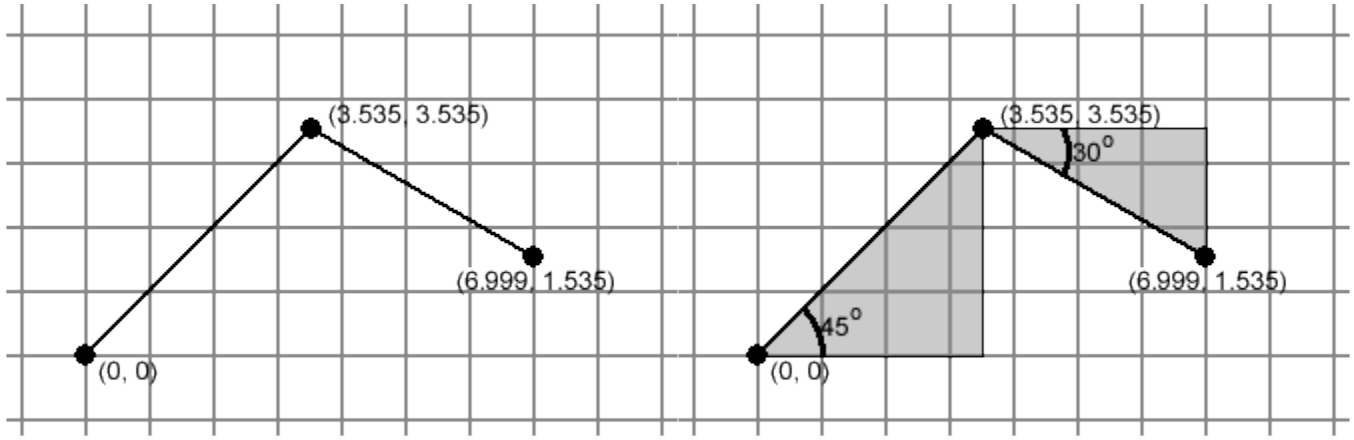
Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Experto	**Universidad**: UPR-Bayamón
Autor: ACSL-1998	**Tipo de competencia**: Programación
Problema #: 4	

Treasure Island

You are at the center of an Island and you'll be given directions to the buried treasure. The directions will be specified as ordered pairs representing the distance (in miles) and the direction (in degrees) to travel. All of the directions will be multiples of 30 or 45 degrees. An angle of 0 means due east, an angle of 90 is due north, whereas an angle of -45 is southeast. Your job is to report the location of the hidden treasure.

For example, suppose you were given the directions (5, 45) and (4, -30). This means to travel 5 miles in the direction of 45 degrees, which takes you to (3.535, 3.535). From there, you head 4 miles in a southeasterly direction, which takes you to (6.999, 1.535).

There are two important formulas to know. The sides of a 45-45-90 triangle are in the proportions 1-1-sqrt(2) and the sides of a 30-60-90 triangle are in the proportions 1-sqrt(3)-2. The following diagram illustrates how these facts were used to solve the sample problem above:



In the first move, the sides of the triangle are 3.535-3.535-5, and in the second move, the sides are 2-3.464-4. (The value of sqrt(2)=1.414 and sqrt(3)=1.732.) You are not allowed to use any trigonometry functions (sin, cos, etc.).

There will be 10 sets of data. Each set consists of a number *N* followed by *N* ordered pairs of (distance, direction). The distance will be a positive number, and the direction will be a multiple of 30 or 45. The ordered pairs specify segments to follow from the origin to the location of buried treasure. For each set of input, print the x,y location of the buried treasure. Your answer must match our answers to within plus or minus 0.001.

As mentioned above, you may not use any trigonometric functions (sin, cos, etc.). Proctors will scan your programs for any use.

Sample Input:

```
Line 1: 2, 5, 45, 4, -30
Line 2: 4, 2, 90, 2, 180, 2, 270, 2, 0
```

Sample Output:

```
Output 1: 6.999, 1.535
Output 2: 0, 0
```

Programming Problem #4 - Treasure Island - Experto

Test Data Input:

```
 #1: 3, 2, 45, 2, 30, 2, 60
 #2: 3, 2, -45, 2, -30, 2, -60
 #3: 3, 2, 90, 2, 180, 2, 270
 #4: 3, 2, -90, 2, -180, 2, -270
 #5: 3, 2, 30, 2,30, 2, 30
 #6: 4, 2, -45, 2, -90, 2, -180, 2, 180
 #7: 4, 2, 360, 2, -360, 2, 720, 2, -720
 #8: 4, 2, -45, 2, -135, 2, -225,  2, -315
 #9: 4, 2, 390, 2, 750, 2, -120, 2, -180
#10: 5, 2, 0, 2, 90, 2, -90, 2, -270, 2, -180
```

Test Data Output:

```
 #1: 4.14626, 4.14626
 #2: 4.14626, -4.14626
 #3: -2, 0
 #4: -2, 0
 #5: 5.19615, 3
 #6: -2.58578, -3.41421
 #7: 8, 0
 #8: 0,0
 #9: .464101, .267949
#10: 0, 2
```

Universidad de Puerto Rico
Bayamón, Puerto Rico

Cuartas Competencias de Programación
2003

Intermedio

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Intermedio	**Universidad**: UPR-Bayamón
Autor: Prof. Nelliud D. Torres	**Tipo de competencia**: Programación
Problema #: 1	

ESTRELLAS

Archivos:

Input: estrellas.dat
Output: N/A

Definición:

Un observatorio astronómico requiere de un programa que analice una fotografía del cielo tomada por la noche. La información de la fotografía está almacenada en forma de tabla, donde cada elemento representa la cantidad de luz que se registró para cada punto. Los valores registrados van del 0 al 20, por ejemplo:

0	3	4	0	0	0	6	8
5	13	6	0	0	0	2	3
2	6	2	7	3	0	10	0
0	0	4	15	4	1	8	0
0	0	7	12	6	9	10	4
5	0	6	10	6	4	8	0

La persona encargada de analizar la información supone que hay una estrella en (***i, j***) si:

- el punto no se encuentra en las orillas de la fotografía (primero o último renglón o columna), y
- ***(a[i, j] + a[i - 1, j] + a[i + 1, j] + a[i, j - 1] + a[i, j + 1]) > 30***

Se espera como resultado del análisis, una tabla ***b*** con un "*" en las parejas (***i, j***) en las que se supone que hay una estrella. El resto de la tabla debe quedar lleno de espacios. La tabla ***b*** que resulta del ejemplo anterior es:

	1	2	3	4	5	6	7	8
1								
2		*						
3								
4				*				
5				*	*		*	
6								

Problema:

Realice un programa en C++ que:

1. Lea las dimensiones de la tabla ***m*** y ***n*** con (***1 ≤ m, n ≤ 20***).
2. Lea los valores de cada elemento de la tabla ***a***.
3. Construya la tabla ***b***.

4. Imprima la tabla ***b***.

Ejemplo de Input:

N/A – Leer la tabla de un archivo

Ejemplo de Output (Screen):

	1	2	3	4	5	6	7	8
1								
2		*						
3								
4				*				
5				*	*		*	
6								

El total de estrellas es 5

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Intermedio	**Universidad**: UPR-Bayamón
Autor: Furman University 2000	**Tipo de competencia**: Programación
Problema #: 2	

Super Freq

Input File Name: **prob2.in**
Output: **to the screen**

Description:

A character is known to its homeboys as a *super freq* if it occurs with frequency strictly greater than 15% in a given passage of text. Write a program that reads an ASCII text file and identifies all the English alphabet (A-Z, a-z) super freqs in that text file. Your program should be case insensitive.

There will only be one passage of text in the input file, and it will consist of a single line.

Use all characters in the file (including spaces and punctuation) in calculating the frequency of a particular alphabet character, but you don't have to compute the frequency of these "extra" characters.

Example:

Input: **Sally sells sea shells by the sea shore.**
Output: **S is a super freq.**

Input: **How now brown cow.**
Output: **O is a super freq.**
W is a super freq.

Input: **Hey Sam!**
Output: **There are no super freqs.**

Additional notes:

The input file will not contain any tabs, carriage returns or other special formatting characters.

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Intermedio	**Universidad**: UPR-Bayamón
Autor: ACSL-1998	**Tipo de competencia**: Programación
Problema #: 3	

Post2In

Given an expression in a functional language expressed in postfix, convert it to infix. The language will consist of the unary function *abs* and the binary functions *min* and *max*. Read each input as a single string; the output should have a space after each comma.

Sample Data (2 sets of data; the test data has 10 sets of data):

Input	Output
`2 -4 min abs 2 1 max max`	`max(abs(min(2, -4)), max(2, 1))`
`2 21 13 max min abs`	`abs(min(2, max(21, 13)))`

Program # 3. Post2In - Intermedio

Input	Output
1 2 max	max(1, 2)
1 2 3 max min	min(1, max(2, 3))
1 2 max 3 min	min(max(1, 2), 3)
11 22 max 33 44 min max	max(max(11, 22), min(33, 44))
0 abs abs	abs(abs(0))
1 -2 min abs	abs(min(1, -2))
1 abs -2 min	min(abs(1), -2)
1 abs 2 abs min	min(abs(1), abs(2))
-1 abs -2 abs min abs	abs(min(abs(-1), abs(-2)))
11 -2 max -33 44 -5 max min min	min(max(11, -2), min(-33, max(44, -5)))

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Intermedio	**Universidad**: UPR-Bayamón
Autor: ACSL-1998	**Tipo de competencia**: Programación
Problema #: 4	

Botchagaloop

The *botchagaloop* value of a number *x* is found as follows. First, convert *x* to base 8. Call this *p*. Next, sort the digits of *p* in increasing order. Call this *q*. Subtract *q* from *p* (in base 8, of course). Repeat the "sort-subtract" sequence 4 more times, or until the digits in the result are in sorted order (whichever come first). Finally, convert the number back to base 10.

For example, 3418 has a botchagaloop value of 1008. It is computed as follows:

3418 = 6532 (base 8);

1. 6532 - 2356 = 4154;
2. 4154 - 1445 = 2507;
3. 2507 - 257 = 2230;
4. 2230 - 223 = 2005;
5. 2005 - 25 = 1760;

and finally, 1760 = 1008 (base 10).

Note that there is at least one subtraction and at most 5 subtractions.

There will be 5 inputs. Each is a positive integer less than 1,000,000. Print the botchagaloop value of each input.

Sample Input:

```
Enter number: 3418
```

Sample Output:

```
The number is: 1008
```

Sample Input:

```
Enter number: 123
```

Sample Output:

```
The number is: 28
```

Universidad de Puerto Rico
Bayamón, Puerto Rico

Cuartas Competencias de Programación
2003

Principiante

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría:Principiante	**Universidad**: UPR-Bayamón
Autor: Prof. Nelliud D. Torres	**Tipo de competencia**: Programación
Problema #: 1	

CONJETURA DE ULLMAN

Archivos:

Input: N/A
Output: N/A

Definición:

La conjetura de Ullman establece que si empiezas con cualquier número positivo, al aplicar el siguiente procedimiento, siempre te va a dar uno:

1. Empezar con cualquier entero positivo.
2. Si es par, dividirlo entre dos.
3. Si es impar se multiplica por tres (3) y se le suma uno(1).
4. Por cada entero que se obtenga que no sea uno (1) se vuelve a repetir los pasos 2 y 3
5. Al final, se debe obtener uno (1).

Problema:

Desarrolle un programa que le solicite al usuario un número entero y comienze a mostrar en pantalla los resultados de los pasos 2 y 3 hasta que llegue a uno.

Ejemplo de corrida:

Entre un número entero positivo: **26**

26 ← Lo divido entre dos
13 ← Multiplico por 3 y le sumo 1
40 ← Lo divido entre dos
20 ← Lo divido entre dos
10 ← Lo divido entre dos
5 ← Multiplico por 3 y le sumo 1
16 ← Lo divido entre dos
8 ← Lo divido entre dos
4 ← Lo divido entre dos
2 ← Lo divido entre dos
1 ← Resultado Final

Importante: Tiene que poner la corrida exactamente como se ve aquí.

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría:Principiante	**Universidad**: UPR-Bayamón
Autor: Prof. Nelliud D. Torres	**Tipo de competencia**: Programación
Problema #: 2	

HISTOGRAMA DE PALABRAS

Archivos:

Input: palabras.txt
Output: N/A

Definición:

Se quiere poder hacer un histograma vertical que indique la cantidad de caracteres que tiene una palabra utilizando asteriscos.

Problema:

Desarrolle un programa que lea palabras de una archivo llamado "palabras.txt" y muestre en pantalla un histograma horizontal de cada palabra. En el archivo cada palabra estará separada por un espacio en blanco.

Ejemplo de Input:

Un archivo que tenga lo siguiente: **Esta es una muestra de archivo**

Ejemplo de Output(Screen):

```
PALABRA     HISTOGRAMA

Esta        ****
es          **
una         ***
muestra     *******
de          **
archivo     *******
```

<u>Importante</u>, ninguna palabra será mayor de 15 caracteres. Cada palabra puede tener letras mayúsculas o minúsculas. Tiene que incluir el encabezamiento como se muestra aquí y la indentación entre la palabra y los asteriscos.

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Principiante	**Universidad**: UPR-Bayamón
Autor: ACSL-1998	**Tipo de competencia**: Programación
Problema #: 3	

Cabracci

The terms of the Fibonacci sequence are the numbers 1, 1, 2, 3, 5, 8, 13, ... The next term in the sequence is found by summing the previous two terms.

Numbers in a *Cabracci* sequence are formed by summing the previous three terms and subtracting 3. For example, the Cabracci sequence starting with 1, 0, and 4 is as follows: 1, 0, 4, 2, 3, 6, 8, 14, 25, ...

There will be 5 sets of data. Each set consists of four integers: a1, a2, a3, and N. The first three numbers are the first three terms of a Cabracci sequence. The fourth number is the term to print. For each input set, print the Nth term of the Cabracci sequence defined by the first three numbers in the input set.

Sample Input: (Screen)

```
Enter 4 numbers: 1 0 4 9
```

Sample Output: (Screen)

```
The number is: 25
```

Sample Input: (Screen)

```
Enter 4 numbers: 1 1 2 5
```

Sample Input: (Screen)

```
The number is: 1
```

Programming Problem #3 – Cabracci - Principiantes

Test Data Input:

```
#1: 1, 1, 1, 10
#2: 1, 3, 5, 12
#3: -1, -2, -3, 10
#4: 2, 4, 6, 8
#5: 5, 10, 15, 6
```

Test Data Output:

```
#1: -51
#2: 670
#3: -386
#4: 91
#5: 88
```

Fecha: 26/abril/2003	**Nombre de la competencia**: 4tas Competencias Interuniversitarias
Categoría: Principiantes	**Universidad**: UPR-Bayamón
Autor: Furman University 1995	**Tipo de competencia**: Programación
Problema #: 4	

Balanced Parentheses

Input: PARENTESIS.TXT
Output: N/A (SCREEN)

Description:

Write a program that checks an arithmetic expression for balanced parentheses. For example, expressions containing the sequences ( ) and
((( ) ( ))) are balanced but ())(and ((()) are not.

The input file contains a series of expressions, one per line; each line is ended with a carriage return. For each expression. decide only if it has balanced parentheses. Echo the expression and indicate whether it WELL-FORMED or NOT WELL-FORMED.

Example:

Input:

(2+3)
(2))
((-5+2)*(4-3))
4/1
)3*2)

Output:

(2+3) IS WELL-FORMED
(2)) IS NOT WELL-FORMED
((-5+2)*(4-3)) IS WELL-FORMED
4/1 IS WELL-FORMED
)3*2) IS NOT WELL-FORMED

Additional notes:

The program does not have to account for the rest of the expression, that is, whether the operators and operands are legal or not. It is required to test only for parentheses.

Universidad de Puerto Rico
Bayamón, Puerto Rico

Terceras Competencias de Programación
2002

Principiante

Fecha: 20/abril/2002	**Nombre de la competencia**:_____
Categoría: Principiante	**Universidad**: UPR- Bayamón
Autor: _____	**Tipo de Competencia**: _____
Problema #:	**Algoritmos**:_____

Common Letters

Problema:

Write a program that takes two words and finds any common letters that they have. For example, the words 'computer' and 'program' have the letters 'o', 'm', 'p', 'r', in common. The output should be both words with all common letters in capitals. Neither word will have more than 8 letters.

Ejemplo de Input:

```
Enter two words : computer program
```

Ejemplo de Output:

```
cOMPuteR PROgRaM
```

Test your program with the following pairs of words

```
hello all
bye all
```

Fecha:20/abril/2002	**Nombre de la competencia**:_____
Categoría:Principiantes	**Universidad**: UPR- Bayamón
Autor: _____	**Tipo de competencia**:_____
Problema #:	**Algoritmos**:_____

String Compression

Problema:

Consider the string 'AAAABCCCCCDDDD' consisting of alphabetic characters only. This string is of length 14. Since the string consist of alphabetic characters only, duplicate characters can be removed and replaced with a duplication factor n. With this technique the string can be compressed and representing by '4AB5C4D'. The compressed string is of length 7. Write a program which takes a string in compressed form and recreates the original uncompressed string.

Ejemplo d Input:

The string will be of the format 'nA…' where n, the duplication factor, is an integer between 2 and 99 and A is an uppercase alphabetic character. A string may contain single characters not prefixed with a duplication factor. If this were not the case, for instance, the string 'AABCDE' would be compressed to 'AABCDE' would be compressed to '2ABCDE'.
The maximum length of an input string is 80 characters.

Ejemplo de Output:

The uncompressed string, 40 characters per line (it may be necessary to break an uncompressed string over multiple lines).

Example 1

Input
```
Entre el string: 3A4B7D
```

Output
```
AAABBBBDDDDDDD
```
Example 2

Input
```
Entre el string: 22D7AC18FGD
```

Output
```
DDDDDDDDDDDDDDDDDDDDDDAAAAAAACFFFFFFFFFF
FFFFFFFFGD
```

Fecha: 20/abril/2002	**Nombre de la competencia**:_____
Categoría: Principiante	**Universidad**: UPR-Bayamón
Autor: Prof. Nelliud D. Torres	**Tipo de Competencia**:_____
Problema #:	**Algoritmos**:_____

DECIMAL COMPLEMENTS

Definición:

Muchas computadoras para representar un número negativo, lo almacenan como su complemento aritmético. Esto permite que al efectuar una resta, se pueda utilizar la suma y obtener el resultado esperado. Para obtener el complemento de un número en nuestro sistema numérico decimal, tenemos primero que calcular el complemento base nueve del número. Esto es, cuantos números faltan para que el dígito se convierta en nueve. Por ejemplo el complemento base nueve de 5 es 4, de 3 es 6, de 3 es 7 de 0 es 9 y así por el estilo. El complemento base diez se obtiene al sumarle uno al complemento base nueve. El procedimiento que ejecuta una computadora para restar mediante la suma es el siguiente:

1. Supongamos que tenemos la siguiente resta: 6142-4816
2. Convertimos el sustraendo 4816 en complemento base nueve lo cual da 5183.
3. Le sumamos uno al resultado: 5183+1=5184
4. Sumamos ambos números 6142+5184=11326
5. Eliminamos el último dígito a la izquierda y obtenemos el resultado: 1326

Problema:

Desarrolle un programa que solicite al usuario una resta. Luego muestre en pantalla todos los pasos necesarios para poder obtener el resultado utilizando el complemento base 10.

Ejemplo de Input:

Entre una resta: **327□-□ 153**

Nota:
□ = Espacio en Blanco

Ejemplo de Output:

Número seleccionado → 153
Complemento base nueve → 846
Complemento base diez → 847
Resultado de la suma → 1174
Resultado → 174

Fecha: 04/20/2002	**Nombre de la competencia**:_____
Categoría: Principiante	**Universidad**: UPR-Bayamón
Autor: Antonio Huertas	**Tipo de Competencia**:_____
Problema #:	**Algoritmos**:_____

BANNER NUMERICO

Problema:

Desarrolle un programa que convierta un dato numérico entrado por pantalla en un "Banner". Sólo se aceptarán números y los símbolos + y -. El tamaño del dígito debe ser de 6 columnas por 5 filas. A continuación un ejemplo de cada dígito.

```
001100    222222    333333    400400    555555    666666
111100    000002    000003    400400    500000    600000
101100    222222    333333    444444    555555    666666
001100    200000    000003    000400    000005    600006
111111    222222    333333    000400    555555    666666


077777    888888    999999    000000    00++00    000000
000007    800008    900009    0         00++00    000000
000007    888888    999999    0         ++++++    ------
000007    800008    000009    0         00++00    000000
000007    888888    999999    000000    00++00    000000
```

El símbolo 0 representa un espacio en blanco.

Ejemplo de Input:

Entre un número: +3578

Ejemplo de Output:

```
++      333333    555555    777777    888888
++           3 5                 7 8      8
++++++  333333    555555         7  888888
++           3         5         7 8      8
++      333333    555555         7  888888
```

Universidad de Puerto Rico
Bayamón, Puerto Rico

Terceras Competencias de Programación
2002

Intermedio

Fecha: 20/abril/2002	Nombre de la Competencia:_____
Categoría: Intermedio	Universidad: UPR- Bayamón
Autor: _____	Tipo de Competencia:_____
Problema #:	Algoritmos:_____

WELL ORDERED NUMBERS

Definición:

The number 138 is called well-ordered because the digits in the number (1,3,8) increase from left to right (1 <3 <8). The number 365 is not well-ordered because 6 is larger than 5.

Problema:

Write a program that will find and display all possible three digit well-ordered numbers. Report the total number of three digit well-ordered numbers.

Ejemplo de Output:

Example

The three digit well ordered numbers are:

```
123   124  125 126  127  128  129  134
135   136  137  138  139  145  146  147
148   149  156  157  158  159  167  168
...   ...  ...  ...  ...  ...  ...  ...
...   ...  ...  ...  ...  ...  ...  ...
678   679  689  789
```

The total number is ??

Fecha: 20/abril/2002	**Nombre de la Competencia**: _____
Categoría: Intermedio	**Universidad**: UPR- Bayamón
Autor:_____	**Tipo de Competencia**: _____
Problema #:	**Algoritmos**:_____

HORIZONTAL HISTOGRAM

Problema:

Write a program that accepts a set of digits (0 to 9) as input and prints a horizontal histogram representing the occurrences of each digit.

Test your program with the set of 13 digits:
1,71,2,9,,7,1,3,7,5,7,9,0

Ejemplo de Input:

```
Enter a Number: 12
Enter 12 digits:
1,7,2,9,6,7,1,3,7,5,7,9
```

Ejemplo de Output:

```
0
1   * *
2   *
3   *
4
5   *
6   *
7   * * * *
8
9   * *
```

Fecha: 20/abril/2002	**Nombre de la Competencia**:_____
Categoría: Intermedio	**Universidad**: UPR- Bayamón
Autor: _____	**Tipo de Competencia**:_____
Problema #:	**Algoritmos**: _____

SHUTTLE PUZZLE

Definición:

The SHUTTLE PUZZLE of size 3 consists of 3 white marbles, 3 black marbles, and a strip of wood with 7 holes. The marbles of the same color are placed in the holes at the opposite ends of the strip, leaving the center hole empty.

```
INITIAL  STATE:  WWW  BBB

GOAL STATE:        BBB    WWW
```

Problema:
To solve the shuttle puzzle use only two of moves. Move 1 marble 1 space (into the empty hole) or jump 1 marble over 1 marble of the opposite color (into the empty hole). You may not back up, and you may not jump over 2 marbles.

A Shuttle Puzzle of size N consists of N white marbles and N black marbles and 2N+1 holes.

Write a program that will solve the SUTTLE PUZZLE for any size N(*10) and display the board after each move. Use W to represent a white marble and B to represent a black marble and a blank to represent the empty hole. Text your program for N=3 and N=4.

Ejemplo de Input:
Indique el largo de N: **3**
Ejemplo de Output:

```
WWW  BBB
WWWB  BB
WW  BWBB
W  WBWBB
WBW  WBB
WBWBW  B
WBWBWB
WBWB  BW
WB  BWBW
  BWBWBW
B  WBWBW
BBW  WBW
BBWBW  W
BBWBW  W
BBWB  WW
BB  BWWW
BBB  WWW
```

Fecha: 20/abril/2002	**Nombre de la competencia**:_____
Categoría: Intermedio	**Universidad**: UPR-Bayamón
Autor: Prof. Nelliud D. Torres	**Tipo de competencia**:_____
Problema #:	**Algoritmos**:_____

NUMBER FANTASIES

Archivos:

Input: N/A
Output: N/A

Definición:

Trabajar con números en ocasiones puede parecer mágico. Se dice que si usted escoge una cifra de tres dígitos cuyas unidades y centenas no sean iguales (números diferentes) y ejecuta el siguiente procedimiento, el resultado siempre debe ser 1089.

1. Seleccione un número de tres dígitos con unidades y centenas diferentes. Ej. 285
2. Invierta los dígitos. Ej. 582
3. Reste los números de modo que el resultado sea positivo. Ej. 582-285=297
4. Invierta los dígitos nuevamente Ej. 792 + 297
5. El resultado siempre debe ser 1089

Problema:

Desarrolle un programa que solicite al usuario una cifra de tres dígitos con unidades y centenas diferentes. Muestre en pantalla los pasos necesarios para convertir el número a 1089.

Ejemplo de Input:

Indique una cifra de 3 dígitos: **97**
Error son tres dígitos
Indique una cifra de 3 dígitos:**121**
Error las unidades y centenas tienen que ser diferentes
Indique una cifra de 3 dígitos: **821**

Ejemplo de Output:

Número seleccionado → 821
Número invertido → 128
Resta → 821-128=693
Invertir el resultado→ 396
Suma → 693 + 396
Resultado → 1089

Universidad de Puerto Rico
Bayamón, Puerto Rico

Terceras Competencias de Programación
2002

Experto

Fecha: 20/abril/2002	**Nombre de la Competencia**: _____
Categoría: Experto	**Universidad**: UPR-Bayamón
Autor:_____	**Tipo de Competencia**: _____
Problema #:	**Algoritmos**:_____

6x6 CHECKER CHALLENGER

Problema:

Examine the 6x6 checkerboard below and note that the six checkers are arranged on the board so that own and only one is placed in each row and each column, and there is never more than one in any diagonal. (Diagonals run from southeast to northwest and southwest to northeast and include all diagonals, not just the major two.)

```
          Column
          1   2   3   4   5   6
        -------------------------------
1       |   |0  |   |   |   |   |   |
        -------------------------------
2       |   |   |   |0  |   |   |   |
        -------------------------------
3       |   |   |   |   |   |   |0  |
        -------------------------------
4       |0  |   |   |   |   |   |   |
        -------------------------------
5       |   |   |0  |   |   |   |   |
        -------------------------------
6       |   |   |   |   |0  |   |   |
        -------------------------------
```

The solution shown above is described by the sequence 2 4 6 1 3 5, which gives the column positions of the checkers for each row from 1 to 6.

```
ROW          1 2 3 4 5 6
COLUMN       2 4 6 1 3 5
```

This is one solution to the 6x6 Checkers Challenge. Write as program that searches and finds all unique solutions sequences to the 6x6 Checkers Challenge. Print out the solution using the column notation described above and count the total number of solutions found (including reflection and rotations).

Ejemplo de Output:

```
2 4 6 1 3 5
3 6 2 5 1 4
? ? ? ? ? ?
? ? ? ? ? ?

THERE ARE  ?  SOLUTIONS TO THE 6X6 CHECKERS CHALLENGE.
```

Fecha: 20/abril.2002	Nombre de la competencia:_____
Categoría: Experto	Universidad: UPR-Bayamón
Autor: Nelliud D. Torres	Tipo de competencia: _____
Problema #:	Algoritmos:_____

NÚMERO OCULTO

Archivos:

Input: OCULTO.INP
Output: N/A

Definición:

De acuerdo a una tabla dada de valores, determine cual es el número oculto. Cada esquema de pistas con las que usted podrá deducir un número compuesto por cuatro cifras distintas (elegidas del 0 al 9), que no empieza con cero. En la columna B (de bien) indicamos cuántos dígitos hay allí en común con el número buscado y en la misma posición. En la columna R (de regular) se indica la cantidad de dígitos en común pero en posición incorrecta. Si en algún caso encuentra tres de los cuatro dígitos que forman el número misterioso y no da con el restante (que no es ninguno de los dígitos que intervienen en los números-pista) deberá buscar cuál es el dígito que no forma parte de dichos números-pista. Si se trata de un único número ausente, ése será el cuarto dígito buscando.

EJEMPLOS:

				B	**R**
9	0	5	2	0	1
6	5	1	0	2	1
5	7	1	0	1	1
7	0	8	9	1	1

El número oculto es: 6180

Problema:

Desarrolle un programa que lea una tabla de un archivo de Input y muestre como Output el número secreto.

Ejemplo de Input:

4 □ 2 □ 5 □ 8 □ 0 □ 1

9 □ 3 □ 7 □ 8 □ 0 □ 2

7 □ 4 □ 1 □ 2 □ 1 □ 2

8 □ 7 □ 3 □ 4 □ 2 □ 0

Nota:
- = Espacio en Blanco

Ejemplo de Output:

EL NÚMERO OCULTO ES: 1732

Ejemplo de otras tablas con sus respuestas:

				B	**R**
1	3	2	7	1	1
6	3	0	7	1	0
7	4	8	5	0	2
4	0	2	5	1	2

EL NUMERO OCULTO ES: 4352

				B	**R**
2	4	5	0	1	0
7	4	3	5	0	0
4	1	5	0	1	0
4	9	8	0	1	1

EL NUMERO OCULTO ES: 2189

				B	R
6	7	5	9	0	2
6	2	7	8	2	1
5	9	3	6	0	1
6	5	8	4	0	1

EL NUMERO OCULTO ES: 7298

				B	R
9	1	0	6	1	1
2	3	8	0	2	0
1	5	7	0	1	0
7	5	9	8	0	2

EL NUMERO OCULTO ES: 9480

Fecha: 20/abril/2002	**Nombre de la competencia:** _____
Categoría: Experto	**Universidad:** UPR-Bayamón
Autor: Prof. Nelliud D. Torres	**Tipo de competencia:** _____
Problema #:	**Algoritmos:** _____

MULTIPLICACIÓN POR EL MÉTODO DE LA REJILLA

Archivos:

Input: N/A
Output: N/A

Definición:

En la Inglaterra del siglo XVI, los estudiantes de matemáticas usaban un sistema de multiplicación que, aunque pueda parecer un poco incómodo, da la respuesta correcta rápidamente. Se llama el método de la rejilla y opera de la siguiente forma.

Supongamos que se desea multiplicar **123** por **456**

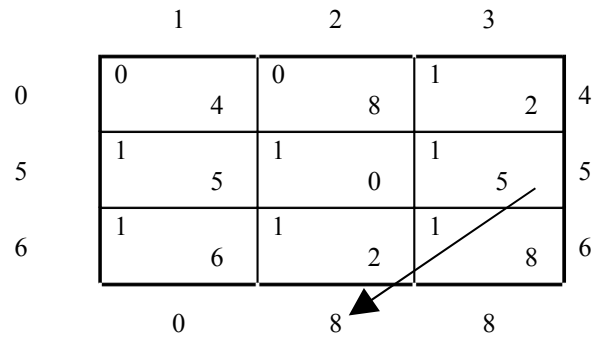
1. Dibujamos una matriz de 3 x 3 y escribimos uno de los números en la parte de arriba y el otro en la parte de abajo. También dividimos diagonalmente cada una de las celdas.

1	2	3	
			4
			5
			6

2. Se multiplica el número final de cifra superior (3) por cada uno de los números de la cifra que está a la derecha. El resultado se almacena en la celda en donde intersecan. Si el resultado de la multiplicación es mayor de nueve, se coloca el segundo dígito en la parte superior diagonal. Su ya había un número en esa posición, se pueden sumar. Esta operación se repite con los otros dos dígitos (1 y2).

1	2	3	
0 / 4	0 / 8	1 / 2	4
0 / 5	1 / 0	1 / 5	5
0 / 6	1 / 2	1 / 8	6

3. Se suma cada una de las diagonales comenzando con la del triángulo inferior derecho y se anotan los totales a lo largo de la parte inferior de derecha a izquierda y en el lateral izquierdo de abajo hacia arriba. Si la suma de una de las diagonales es mayor de 10, se anota la cifra de las unidades en su lugar correspondiente y la de las decenas en la diagonal siguiente.



4. Escribiendo las cifras nos da como resultado: **056088 = 56,088**

Problema:

1. Desarrolle un programa que acepte dos números de un largo no menor de dos ni mayor de cinco.
2. Tiene que mostrarse en pantalla el proceso de multiplicación tal y como se expuso anteriormente paso por paso. No tiene que ser en un formato gráfico.
3. Finalmente tiene que mostrar el resultado obtenido.

Ejemplo de Input:

Indique el primer dígito: **584**
Indique el segundo dígito: **4173**

Ejemplo de Output:

```
584
22 / 03 / 21 /64
41 / 50 / 80  / 41
33 / 55 / 62 /87
73 /53 / 41 / 23
032

Resultado = 2437032
```

Nota:
= Espacio en Blanco

Fecha: 04/20/2002	**Nombre de la competencia:** _____
Categoría: Experto	**Universidad:** UPR-Bayamón
Autor: Prof. Juan M. Solá	**Tipo de competencia:** _____
Problema #:	**Algoritmos:** _____

HTML CODE OPTIMIZER

Algunos editores de páginas de Internet colocan tags de HTML redundantes. Inclusive existen tags con contenido vacío. Ejemplo: <P> </P>. Este es un tag que abre y cierra un párrafo con un espacio adentro. Otro posible problema es que encontremos un párrafo que abre y cierra y que sólo tiene espacios, tabs y enter. Dentro de HTML existen tags redundantes. Por ejemplo en una tabla tenemos tags de la siguiente manera: <TABLE> <TD>1. contenido</TD><TR><TD>2. Contenido</TD></TABLE>. El tag </TD>es redundante ya que no hace diferencia si está o no.

HTML no considera los enters no los espacios múltiples. Para HTML, n espacios = 1 espacio. Los enters y tabs no existen para HTML.

Su programa debe leer un archivo de texto con contenido en HTML y debe eliminar los siguientes tags redundantes: </TD>,</OPTION>, </UL>, </TH> y </LI>. Además debe eliminar los tags que tengan espacios, tabs y enter solamente (por ejemplo: <P> </P>). Si su programa encuentra más de un espacio en blanco, enter o tab, eliminará el exceso. Su programa también cambiará todas las ocurrencias de los siguientes patrones mnemónicos (por ejemplo: &ntilde) por patrones numéricos (por ejemplo: &#241). Vea la siguiente tabla de equivalencias:

INPUT EN HTML	OUTPUT EN HTML	CARÁCTER
Ñ , ñ	Ñ ñ	Ñ ñ
Á á	Á á	Á á
É é	É é	É é
Í í	Í í	Í í
Ó ó	Ó ó	Ó ó
Ú ú	Ú ú	Ú ú

Ejemplo:

Input:

```
<HTML><TITLE>Ejemplo<TITLE>
<P> </P>
<P>Esto es un párrafo

        </P>
<TABLE><TH> Título de la tabla</TH>
<TD>&Ntilde;ame    Bojot&uacute;</TD> </TABLE>

</HTML>
```

Output:

```
<HTML><TITLE>Ejemplo<TITLE>
<P>Esto es un párrafo</P>
<TABLE><TH> Título de la tabla</TR>
<TD>&#209;AME Bojot&#250</TABLE>
</HTML>
```

Universidad de Puerto Rico
Bayamón, Puerto Rico

Primeras Competencias de Programación 2000

Experto

Categoría: EXPERTO

UPRB Programming Contest
Problem set

Autor Nelliud D. Torres

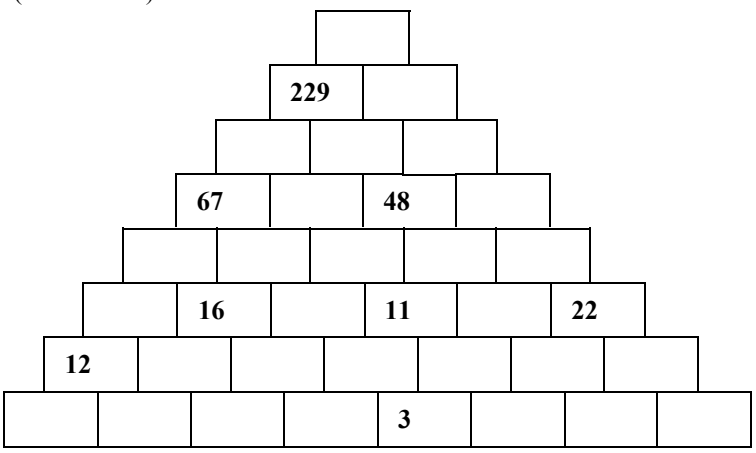
April 28, 2000

PIRÁMIDES NUMÉRICAS

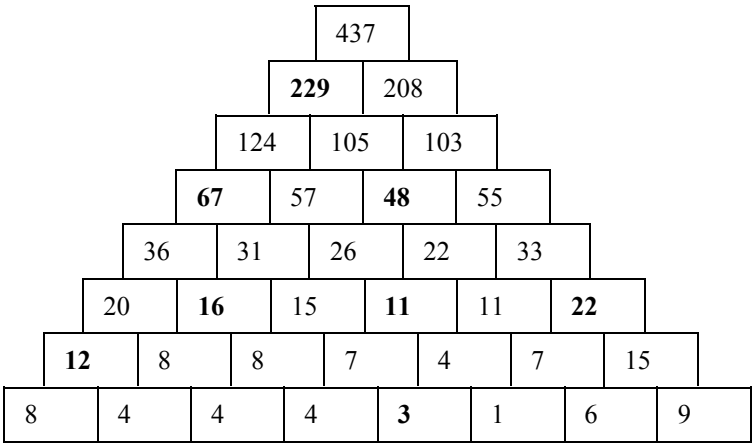
Desarrolle un programa que complete una pirámide colocando un número de una o más cifras en cada casilla, de modo tal que cada casilla contenga las sumas de los dos números de las casilla inferiores. Dentro de la pirámide se establecerán unos números guías que ayudarán a resolver el problema. Los puntos a considerar son los siguientes:

1. Se indicarán ocho(8) números por pirámide.
2. La pirámide tiene ocho(8) filas.
3. Para cada fila podría haber hasta un máximo de tres números pre-establecidos.
4. No todas las filas tienen que tener un número establecido.
5. Podrían haber hasta un máximo de tres (3) filas completamente en blanco por pirámide.

Ejemplo: **Pirámide Inicial** (Pirámide.in)



<u>Pirámide Final</u> (Pirámide.out)



Input:

```
0
229 0
0 0 0
67 0 48 0
0 0 0 0 0
0 16 0 11 0 22
12 0 0 0 0 0 0
0 0 0 0 3 0 0 0
```

Output:

```
437
229 208
124 105 103
67 57 48 55
36 31 26 22 33
20 16 15 11 11 22
12 8 8 7 4 7 15
8 4 4 4 3 1 6 9
```

Categoría: EXPERTO	UPRB Programming Contest Problem set
Autor Nelliud D. Torres	April 28, 2000

LA AMENAZA

En un tablero de ajedrez se colocarán 5 piezas representadas con las siguientes letras: J,K,L,M y N. Estas letras representan un rey, una reina, una torre, un alfil y un caballo aunque no necesariamente en programa a crear debe descubrir cual pieza representa cada letra. Los puntos a considerar son los siguientes:

1. Pueden haber de dos a tres casillas con el valor cero.
2. Siempre habrá una solución.
3. Casilla vacía se representará con un asterisco en el archivo de entrada.
4. Las letras estarán en el archivo en mayúsculas.
5. Las piezas son de un solo color; por lo tanto pueden estar pegadas unas de las otras.

Ejemplo: **Posición en el tablero**

J				**0**			
K							
		0					
L		**M**		**N**	**0**		

Resultado:

J=Rey; **K**=Dama; **L**=Torre; **M**=Caballo; **N**=Alfil

Input: (Tablero.in)

```
*  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *
J  *  *  *  0  *  *  *
*  *  *  *  *  *  *  *
K  *  *  *  *  *  *  *
*  *  0  *  *  *  *  *
L  *  M  *  N  0  *  *
```

Output: (Tablero.out)

J=Rey; **K**=Dama; **L**=Torre; **M**=Caballo; **N**=Alfil

Categoría: EXPERTO

UPRB Programming Contest
Problem set

Author: Juan M. Solá Sloan

April 28, 2000

DNS Cache &

Address Resolution Simulation

Los DNS (Domain Name Servers) manejan las direcciones dadas de acuerdo a la clasificación del sitio pedido. En un WAN las redes se distribuyen por subredes y esas subredes pueden tener otras subredes. En una dirección como esta: lab110apc.cub.upr.clu.edu, EDU es la clasificación del site, CLU es subnet de EDU, UPR es subnet de CLU, CUB es un subnet de UPR y el lab110apc es un subnet de CUB. Cada dirección se convierte en un *octet* (un grupo de 4 n números que representan la dirección real) parecido a este: 217.771.56.1.

Los DNS poseen un árbol con todas las direcciones que existen en su área. Ellos se encargan de traducir la dirección en palabras a una dirección numérica y establecer la conección. Este proceso se llama resolución de una dirección (address resolution). Los DNS utilizan toda su memoria como un inmenso cache. Las direcciones más solicitadas se buscan primero en cache y de no estar se buscan en el árbol.

Usted creará un programa que leerá dos archivos de entrada, uno con las ramas del árbol y otro con los pedidos al DNS (ver ejemplo). Creará un cache para guardar las direcciones más solicitadas al DNS y generará un archivo de salida. (**El árbol tendrá un máximo de 5 ramas. El tamaño del cache será el primer número que aparezca en request.in y puede ser de 0 – 2048 direcciones.)**

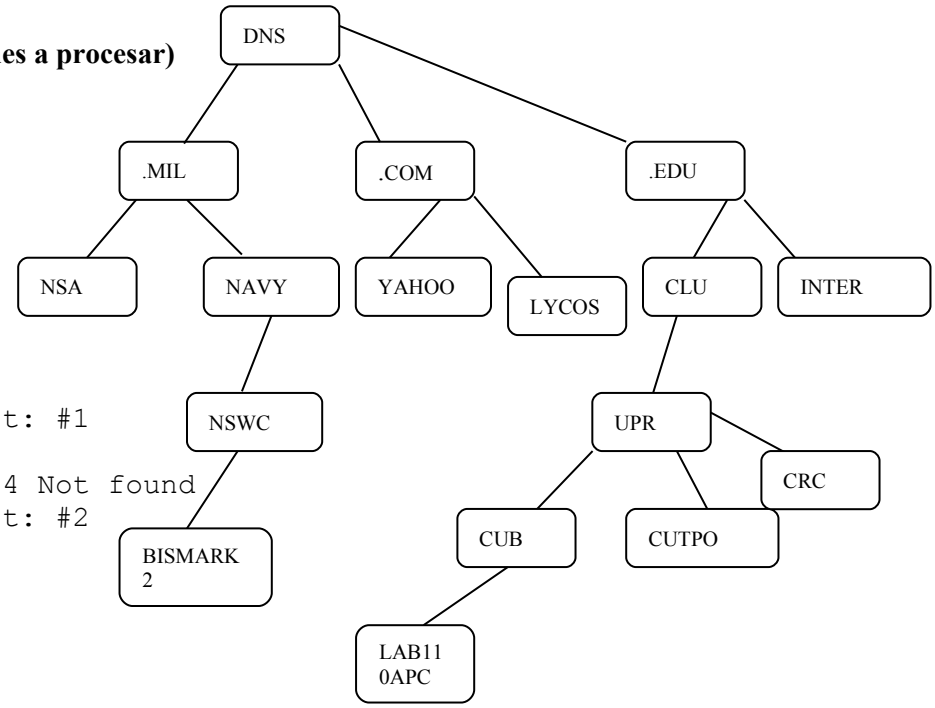
Archivos de entrada:

Database.in (Contiene la estructura del árbol)

.EDU /INTER /CLU //UPR ///CUB ////lab110apc ///CRC ///CUTPO
.MIL /NSA /NAVY //NSWC ///BISMARK_2
.COM /YAHOO /LYCOS

Request.in (Queue de las direcciones a procesar)

```
256
CUB.upr.clu.edu
CUB.upr.clu.edu
Yahoo.com
Yahoo.edu
Marcos.lab12.prtc.net
CUB.upr.clu.edu
```



OUTPUT.OUT

```
Cub.upr.clu.edu  Depth: 4
Cub.upr.clu.edu  cache hit: #1
Yahoo.com        Depth: 2
Yahoo.edu        Error 404 Not found
CUB.upr.clu.edu  cache hit: #2
```

Categoría: Experto

UPRB Programming Contest
Problem set
April 28, 2000

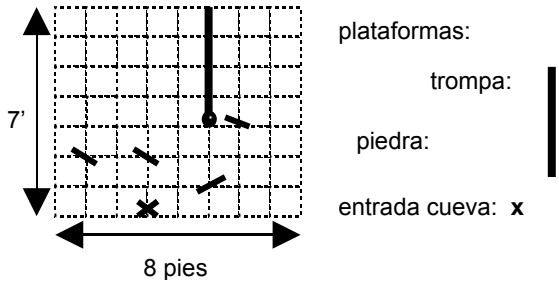
Autor: Hector y Ricardo Martínez

Programa: prog3. xxx
archivo de entrada: prog3.in

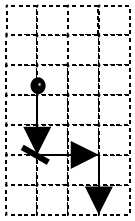
archivo de salida: prog3. out

Problema #3: El elefante Furioso

Un elefante está muy furioso porque unas ratas han construido una cueva en el fondo de su pozo de agua fresca. Por lo que este ha decidido meter la trompa en el agua y tapar la entrada de la cueva con una de las piedras que se encuentran por el área. Desafortunadamente, elefante se encontró con dos problemas: primero, como el pozo es muy profundo no alcanza la entrada, por lo que tendrá que soltar la piedra, y segundo, que no la puede soltar directamente, porque las ratas, cansadas de tanto bucear, han construido ademas unas plataformas inclinadas en las que hacen escala al entrar y salir del agua.



Cuando el elefante suelta la piedra, está acumulada velocidad, y rebota cuando choca con una de las plataformas. Verdaderamente, cada vez que la piedra cae, esta acumulada velocidad. Por otro lado, la pierde cuando sube o cuando se mueve para la izquierda o la derecha. Observe el siguiente ejemplo:



La piedra cae 2',
choca con la plataforma, se desplaza 2' a la derecha,
y cuando pierde toda la velocidad, vuelve a caer.

La piedra puede chocar con las plataformas por encima o por debajo.
El efecto es siempre el mismo: la piedra puede subir o moverse para los lados la misma cantidad de pies que baja.

Problema: Dado que el elefante puede hundir la trompa en cualquier parte del pozo unos 4 pies, determine de donde debe soltar la piedra para que esta tape la entrada de la cueva.

Asumir:

1. El pozo mide 7 pies de ancho por 7 pies de profundidad.
2. El elefante puede mover el último pie de la trompa, por lo que puede, dado que no se encuentre ninguna plataforma en el camino, hundir la trompa 3 pies y 1 para la izquierda o la derecha.
3. El archivo contiene un mapa del pozo, donde los puntos (.) llenan los espacios de agua, las plataformas se representan con 'l' o "/', dependiendo de la dirección de esta, y la entrada de la cueva se marca con una x. La entrada de la cueva siempre esta en el fondo.
4. Si la piedra pierde toda la velocidad, y se encuentra posada sobre el piso o una plataforma, no se mueve de ahí.
5. Si la piedra choca contra una de las paredes, esta rebota en la dirección contraria.
6. Se garantiza que el mapa tiene solución.
7. El girar la trompa se debe especificar como a la izquierda, derecha, o no es necesario.

Data de ejemplo:

```
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . \ .
. . . . . . . .
. . . \ . . . .
. . . . . / . .
. . . X . . . .
```

salida:

distancia del borde: 5’
profundidad: 4’
girar: no es necesario

Fecha: 16 / Mayo / 2000	**Nombre de la competencia**: _____
Categoría: _____	**Universidad**: Univ. De Puerto Rico, Recinto de Bayamón
Autor: Iván Jiménez	**Tipo de competencia**: _____
Problema #: 1	**Algoritmos**: _____

Input File: dblog.in
Output File: dblog.out
Source File: dblog.<xxx>

Database Logging Tables

Problem Description

Most databases have a logging system that keeps track of all data modifications. The log generated by this system aids in the recovery of the database after a crash.

The data modifications are made through concurrent transactions that affect the database. A transaction reads data, UPDATEs the data and saves it in a space in temporary memory, called a page. The transaction requests that all data updates it has made are COMMITted (i.e. written to disk). When all that data is written to disk, the transaction has ENDed. In case of a database, system or disk error it is possible to ABORT a transaction. The logging system keeps a chronological list of these actions. Each list record is of the form:

LSN	Type	TransID	pageID

Each record has a unique log sequence number (LSN). The type is the action being executed by the transaction (update, commit, end or abort). The *transID* is the identification number for the transaction. The *pageID* is the identification number for the page being modified.

To provide a static image of the database state before a crash, the logging system also keeps track of active transactions and modified pages. Two tables are used for this purpose:

- A dirty page table (DPT) keeps track of the pages in use. It stores the *pageID* of the modified page and the *LSN*, called *recLSN*, of the first log record that caused the age to become dirty. Once the transaction that causes the page to become dirty ends or is aborted, the page record in the table is erased.

- A transaction table (TT) contains one entry for each active transaction. The entry contains the *transID,* the *LSN* of the most recent log record for this transaction (called *lastLSN*) and the *status* of the transaction that is in process (active, aborted or commited). Once the transaction is ended or aborted its record in the table is erased.

  Given the log records, reconstruct the dirty page and transaction tables. Show the process that reconstructs the tables. Input file dblog.in contains the log records in the format:

LSN type: transID pageID

The output file dblog.out should contain the dirty page table and transaction table in the format:

LSN: DPT=[pageID, recLSN),…], TT=[(transID, lastLSN, status) ,…]

Notes

- A transaction is active if it has made an update and is not commited, aborted or ended.
- Only update records require *pageIDs*
- A page is not taken off the dirty page table until all transactions that modified it are either commited/ended or aborted.

Sample Input

10 update: T1 P1
20 commit: T1
30 end: T1
40 update: T2 P2
50 update: T3 P3
60 update: T2 P3
70 abort: T2
80 update: T4 P5
90 update: T4 P4
100 commit: T3

Sample Output

10: DPT= [(P1, 10)] , TT= [(T1, 10, a)]
20: DPT= [(P1, 10)] , TT= [(T1, 20, c)]
30: DPT= [ ], TT= [ ]
40: DPT= [(P2, 40)] , TT=[(T2, 40, a)]
50: DPT= [(P2, 40)] , (P3, 50)] , TT= [(T2, 40, a) , (T3, 50, a)]
60: DPT= [(P2, 40)] , (P3, 50)] , TT= [(T2, 60, a) , (T3, 50, a)]
70: DPT= [(P2, 40)] , (P3, 50)] , TT= [(T2, 60, b) , (T3, 50, a)]
80: DPT= [(P3, 50)] , (P5, 80)] , TT= [(T3, 50, a) , (T4, 80, a)]
90: DPT= [(P3, 50)] , (P5, 80) , (P4, 90)] , TT= [(T3, 50, a) , (T4, 90, a)]
100: DPT= [(P3, 50)] , (P5, 80) , (P4, 90)] , TT= [(T3, 100, c) , (T4, 90, a)

]

Fecha: 16 / Mayo / 2000	**Nombre de la competencia**: _____
Categoría: _____	**Universidad**: Univ. De Puerto Rico, Recinto de Bayamón
Autor: Iván Jiménez	**Tipo de competencia**: _____
Problema #: 2	**Algoritmos**: _____

Input File: subset.in
Output File: subset.out

Source File: subset.<xxx>

Subsets

Problem Description

Given a set of characters, find all its subsets. The input file contains a list of characters separated by spaces. The output will contain all the possible subsets; one subset per line.

Sample Input

```
a  b  c
```

Sample Output

```
{ a  b  c  d }
{ a  b  c }
{ a  b  d }
{ a  c  d }
{ b  c  d }
{ a  b }
{ a  c }
{ a  d }
{ b  c }
{ b  d }
{ c  d }
{ a }
{ b }
{ c }
{ d }
{ }
```

Fecha: 16 / Mayo / 2000	**Nombre de la competencia**: _____
Categoría: _____	**Universidad**: Univ. De Puerto Rico, Recinto de Bayamón
Autor: Iván Jiménez	**Tipo de competencia**: _____
Problema #: 3	**Algoritmos**: _____

Word Puzzle ++

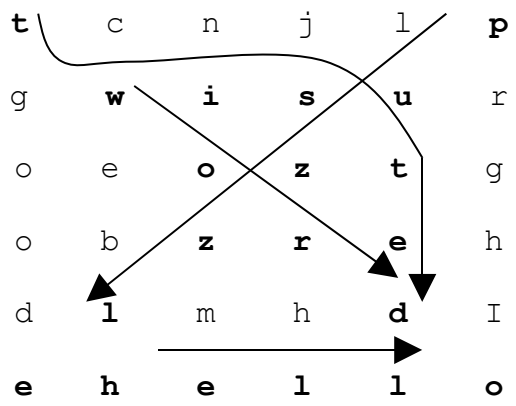
Input File: puzzle2.in
Output File: puzzle2.out
Source File: puzzle2.<xxx>

Puzzle Description

In a word puzzle you are given a list of words and a grid of letters. The objective of the game is to find all listed words embedded in the grid of letters. The words can be found horizontally, vertically or diagonally. In this variation of the game, words can also appear twisted (see figure below).

Make a program that will solve a word puzzle. Your program must find all words present and output the position of each letter of the word on the grid. The input file consists of the dimensions of the grid, the grid of letters and a list of words to find.

The output file will be the list of letters, followed by the coordinate of every letter of the word, or the string "not found".



Sample Input

6
TCHNLP
GWISUR
OEOZTG
OBZREH
DLMHDI
EHELLO
Puzzle

word
problem
hello
twisted
array
input
output

Sample Output

puzzle (0, 5) (1, 4) (2, 3) (3, 2) (4, 1) (5, 0)
word (1, 1) (2, 2) (3, 3) (4, 4)
problem not found
hello (5, 1) (1, 1) (5, 2) (5, 3) (5, 4)
twisted (0, 0) (1, 1) (1, 2) (1, 3) (2, 4) (3, 4) (4, 4)
array not found
input not found
output not found

The output file consists of the numeric sequence followed by the list of names that matched the sequence or the message “No Matches Found”.

Sample Input

```
10
Velez, Carlos
Torres, Ana
Jolie, Angelina
Lopez, Maribel
Laguerre, Tony
Loperena, Martha
Santos, Benjamín
Korg, James
Klinger, Heidi
Questell, Eduardo
4
2345
567
56737
5
```

Sample Output

```
Sequence: 2345
Result:
No Matches Found

Sequence: 567
Results:
Lopez, Maribel
Loperena, Martha
Korg, James

Sequence: 56737
Results:
Loperena, Martha

Sequence: 5
Results:
Jolie, Angelina
Lopez, Maribel
Laguerre, Tony
Loperena, Martha
Santos, Benjamín
Korg, James
Klinger, Heidi
```

Universidad de Puerto Rico
Bayamón, Puerto Rico

Competencias de Programación 1999

Experto

Categoría: Experto	CUB Programming Contest Problem Set
Author: Prof. Juan Solá Sloan	May 29, 1999

Terran vs. Zergs

Terran Message Decypher

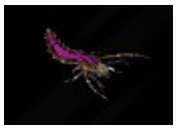
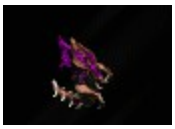
Un ghost está explorando el sector Alpha 4 del planeta X en pársec 26. Hace días que los Terrans no saben de su paradero hasta que un mensaje en clave les llegó.

Decifre el mensaje de los Terrans utilizando de *private key* el valor real 14.25. El primer valor que se recibe en el archivo es el *public key*. Con el public key se obtiene la primera letra del alfabeto en mayúscula (A). El espacio y el punto son las dos últimas letras el protocolo (luego de la Z). La fórmula para decifrar el mensaje es calculada utilizando el valor enviado y el *private key*. La tabla para decifrar cada valor resultante se determina utilizando el public key de base.

Nota: En los mensajes los terrans no están utilizando en este protocolo números. (Quiere decir que si quieren decir 14 lo envían: CATORCE).

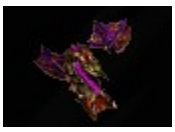
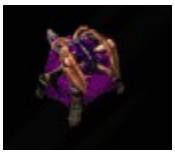
Programe un decifrador del código enviado por el ghost. Utilizara de input el archivo dec.enc y presentará el *output* en pantalla.

Ejemplo:

Input:
134
2265.75 1966.50 2151.75 1995.00

Output:
Z E R G

Categoria: EXPERTO	CUB Programming Contest Problem set
Author: USACO	May 29, 1999

SUPERPRIME RIB

Butchering Farmer John's cows always yields the best prime rib. You can tell prime ribs by looking at the digits lovingly stamped across them, one by one, by FJ and the USDA. Farmer John ensures that a purchaser of his prime ribs gets really prime ribsbecause when sliced from the right, the numbers on the ribs continue to stay prime right down to the last rib, e.g.:

7 3 3 1

The set of ribs 7331 is prime; the three ribs 733 are prime; the two ribs 73 are prime, and, of course, the last rib, 7, is prime. The number 7331 is called a superprime of length 4.

TECHNICAL CONSTRAINTS:

1. For the purposes of prime ribs, the number 1 (by itself) is not a prime number.

2. The number of ribs N satisfies 1²N²8.

3. Exit if the number of ribs is entered as 0.

Write a program that accepts a number of ribs (1..8) and then prints all the superprimes of that length. Exit if the number of ribs entered is 0.

EXAMPLE:

Number of digits: 4

2333 2339 2393 2399 2579 2939 3119 3137 3733 3739 3793 3797 5939 7193 7331 7333 7393

TEST CASES:

N = 7
N = 8

Categoria: EXPERTO	CUB Programming Contest Problem set
Author: USACO	May 29,1999

NUMBER TRIANGLES

```
      7
     3   8
    8   1   0
   2   7   4   4
  4   5   2   6   5
```

Fig. 1

Figure 1 shows a number triangle. Write a program that calculates the highest sum of numbers passed on a route that starts at the top and ends somewhere on the base. Each step can go either diagonally down to the left or diagonally down to the right.

In the sample shown in Fig. 1, the route from 7 to 3 to 8 to 7 to 5 produces the highest sum: 30.

TECHNICAL CONSTRAINTS:

1. Put your input file for each test case in a text file named NUMBER.IN.

2. The number of rows in the triangle is > 1 but <= 100.

3. The numbers in the triangle are all integers between 0 and 99 inclusive.

SAMPLE RUN

NUMBER.IN appears as follows: The first number is the number of rows in the triangle followed by the numbers in each row. Thus the triangle in Fig 1 should be stored in **NUMBER.IN** as follows:

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

The answer is: **<u>30</u>**

TEST CASES:
A test case will be supplied by your teacher or coordinator.

SOLUCIONES

2. Number Triangles

NUMBER.IN

```
15
74
67 51
64 47 54
69 97 1 31
89 14 11 3 72
68 21 70 46 87 68
 5  6  6 29 51 40 71
18 38 15 10 53 19 55 7
 9 70 16 78 89 66 84 7 83
80 19 22 82 19 40 44 35 44 35
85 54 22 16 92 43 21 59 25 18 94
30 19 19  2 33 54 93 95 76 66 43 46
93 36 12 43 26 47 55 26 54 27 23 11 67
51 86 88 42 37 92 61 36 25 18 24 17 16 69
 6 14 10 61 86 52 79 37 96 83 47 44 83 88 20
```

NUMBER.OUT

976

A good program should be able to solve the case where the number of rows is 100 nearly as fast as this case.

If possible, ask the student to generate a random triangle with 100 rows and see if his/her program will still run within the time limit.

A program that tries all possible paths going forward will never make it. It must work backwards.

Superprime Rib

N = 7

2339933
2399333
2939999
3733799
5939333
7393913
7393931
7393933

N = 8

23399339
29399999
37337999
59393339
73939133

ZERO SUM

Consider the sequence of digits from 1 through N (where N=9) in increasing order 1 2 3 4 5 . . . N and insert either a (+) for addition or a (-) for subtraction or a ( ) [blank] to run the digits together. Now sum the result and see if you get zero.

Write a program that will find all sequences of length N that produce a ZERO SUM.

Test Case 1

Input
7
Output
1 + 2 - 3 + 4 - 5 - 6 + 7 = 0
1 + 2 - 3 - 4 + 5 + 6 - 7 = 0
1 - 2 + 3 + 4 - 5 + 6 - 7 = 0
1 - 2 - 3 - 4 - 5 + 6 + 7 = 0
1 - 23 - 45 + 67 = 0
1 - 23 - 45 + 67 = 0

Test Case 2
1 + 2 + 3 + 4 – 5 – 6 – 7 + 8 = 0
1 + 2 + 3 – 4 + 5 – 6 + 7 – 8 = 0
1 + 2 – 3 + 4 + 5 + 6 – 7 – 8 = 0
1 + 2 – 3 – 4 – 5 – 6 + 7 + 8 = 0
1 + 23 – 45 + 6 + 7 + 8 = 0
1 – 2 + 3 – 4 –5 + 6 – 7 + 8 = 0
1 – 2 - 3 + 4 + 5 - 6- 7 + 8 = 0
1 – 2 – 3 + 4 – 5 + 6 + 7 – 8 = 0
1 – 23 – 4 + 5 + 6 + 7 + 8 = 0
12 – 34 – 56 + 78 = 0

You may test this program by entering the integer from the keyboard.
(Use one report form for each student).

Fecha: ____/____/1999	**Nombre de la competencia**: _____
Categoría Experto	**Universidad: CUB**_____
Autor: _____	**Tipo de competencia**:
Problema #: _____	**Algoritmos**: _____

FRIDAY THE 13TH

Is Friday the 13th really an unusual event? That is, does the 13th of the month land on a Friday less often than on any other day of the week? To answer this question, write a program that will compute the frequency that the 13th of each month lands on Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday over a given period of N years. The time period to test will be from January 1, 1990 to December 31, 1900+N-1 for any number of years N. There are a few facts you need to know before you can solve this problem:

1. January 1, 1900 was on a Monday.

2. Thirty days has September, April, June, and November, all the rest have 31 except for February which has 28 except in leap years when it has 29.

3. Every year evenly divisible by 4 is a leap year (1992 = 4*498 so 1992 will be a leap year, but the year 1900 is not a leap year)

4. Rule 3 does not hold for century years. Century years divisible by 400 are leap years, all other are not. Thus the century years 1700, 1800, 1900 and 2100 are not leap years, but 2000 is a leap year.

Test your program for N=20 and N=400.

NOTE: To make it fair for everyone, you may not use any built-in date functions in your computer language.

TEST CASE 1 Enter the number of years N? **<u>20</u>**
FROM JAN 1, 1900 TO DECEMBER 31, 1919 THE 13TH OF THE MONTH LANDS ON

DAY	**# OF TIMES**
SUNDAY	33
MONDAY	**34**
TUESDAY	33
WEDNESDAY	35
THURSDAY	35
FRIDAY	34
SATURDAY	36

(If a solution increments day by day instead of month by month it will take 30 times longer to examine 400 years and might run too long.)

Universidad de Puerto Rico
Bayamón, Puerto Rico

Competencias de Programación 1999

Intermedio

Categoria: INTERMEDIO	CUB Prograrmming Contest Problem set
Author: Nelliud D. Torres	May 29, 1999

PROGRAM LISTING

Una cualidad que parece disminuir con el uso de los lenguajes de programación en ambiente de **PC** es la impresión en papel de un "source program" que pueda listar el código con enumeración por línea y encabezamiento en cada página. Esta opción se usa mucho en ambientes de "Mainframe" con listados de programas largos que se imprimen con la opción "**LISTING"** para poder examinar detenidamente el programa, sus variables e incluso los mensajes de errores.

La opción de "**LISTING**" no solo genera un listado con encabezamiento por página y enumeración por línea, sino que también lista los errores de sintaxis si es que existe alguno y en que línea se encuentra. También puede generar un listado de variables, en donde se definen, en donde se modifican y en donde se utilizan. Esto era de gran ayuda a los programadores de ambiente "Mainframe" y también considero que puede ser de ayuda a los programadores de ambiente **PC**.

PROBLEMA

Haga un programa que lea un programa de Pascal de entrada y genere de salida un encabezamiento por página y enumere una línea de código que tenga el programa.

PUNTOS IMPORTANTES

Tenga en mente lo siguiente

1. El programa debe imprimirse hasta 55 líneas por página.
2. En la primera parte del encabezamiento se utilizarára el nombre del programa puesto después de **PROGRAM** con la extención **PAS**.
3. En la segunda parte se incluirála fecha y hora de la computadora y el día de la semana.
4. Finalmente en la tercera parte se va enumerando las páginas.
5. Después de enumerar en cada línea, se pone el caracter "colon" (:) y se deja por lo menos en un espacio en blanco.
6. En una última página se va a generar un listado de variables por orden alfabético.

EJEMPLO DE CORRIDA

Utilizando de ejemplo el siguiente código:

```
PROGRAM newton (input, output);

CONST
     Epsilon = le-6;

VAR
     Number, root, sqroot: real;

BEGIN
     REPEAT
          writeln;
          write('Enter new number (0 to quit): ');
          read (number);

          IF number = 0 THEN BEGIN
               Writeln(number:12:6, 0.0:12:6);

          END
          ELSE IF number < 0 THEN BEGIN
               Writeln('***ERROR:  number < 0');

          END
          ELSE BEGIN
               Sqroot := sqrt(number);
               writeln(number: 12:6, sqroot: 12:6);
               writeln;

               root := 1;
               REPEAT
                    root :=(number/root + root)/2;
                    writeln(root:24:6,
                         100*abs(root - sqroot)/sqroot:12:2,
                         '%')
               UNTIL abs(number/sqr(root) - 1) <epsilon;
          END
     UNTIL number = 0
END.
```

Se debe generar el siguiente listado:

newton.pas Sat May 29, 1999 10:13:25 Page: 1

```
 1: PROGRAM newton (input, output);
 2:
 3: CONST
 4:            Epsilon = le-6;
 5:
 6:  VAR
 7:            Number, root, sqroot : real;
 8:
 9:  BEGIN
10:       REPEAT
11:             writeln;
12:             write('Enter new number (0 to quit): ');
13:             read(number);
14:
15:             IF number =0 THEN BEGIN
16:                    Writeln(number:12:6, 0.0:12:6);
17:             END
18:             ELSE IF number <0 THEN BEGIN
19:                  Writeln('***ERROR:  number <0');
20:             END
21:             ELSE BEGIN
22:                  Sqroot :=sqrt(number);
23:                  writeln(number:12:6, sqroot: 12:6):
24:                  writeln;
25:
26:                  root := 1;
27:                  REPEAT
28:                      root := (number/root + root)/2;
29:                      writeln(root:24:6,
30:                              100*abs(root - sqroot)/sqroot:12:2,
31:                              '%')
32:                  UNTIL abs(number/sqr(root) - 1) <epsilon;
33:              END
34:       UNTIL number = 0
35:  END.
```

CROSS REFERENCE

```
epsilon         0004 0032
number     0007 0013 0015 0016 0018 0022 0023 0028 0032 0034
root       0007 0022 0023 0026 0028 0028 0028 0029 0030 0032
sqroot          0007 0030 0030
```

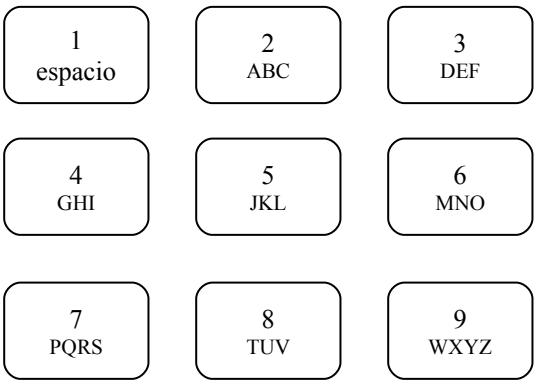
Fecha: 29 / Mayo / 1999	**Nombre de la competencia**: _____
Categoría: Intermedio	**Universidad**: Univ. De Puerto Rico, Recinto de Bayamón
Autor: Iván Jiménez	**Tipo de competencia**: _____
Problema #:	**Algoritmos**: _____

Telephone Directory Search

Problem Description

Some voice mail systems allow users to search for phone numbers of other registered users of the system. To do so, one must spell the name or a prefix of the name of the person to call. This is accomplished by pressing the key number corresponding to each letter of the prefix.

Use the following diagram to map numbers on the keypad to letters.



Write a program that finds all names in the directory that match a given name prefix. The prefix is given as a sequence of keypad presses. Begin by matching the last name and then the first name.

The input will consist of:

- Number of names/records in the dictionary
- List of names in the dictionary
- Number of keypad numeric sequences
- Numeric sequences (one per line).

The input will consist of a file named: **Tel.in**

The output consist of the mumeric sequence followed by the list of names that matched the sequence or the message “No Matches Found”.

Sample Input

10
Velez, Carlos
Torres, Ana
Jolie, Angelina
Lopez, Maribel
Laguerre, Tony
Loperena, Martha
Santos, Benjamín
Korg, James
Klinger, Heidi
Questell, Eduardo
4
2345
567
56737
5

Sample Output

Sequence: 2345
Result:
No Matches Found

Sequence: 567
Results:
Lopez, Maribel
Loperena, Martha
Korg, James

Sequence: 56737
Results:
Loperena, Martha

Sequence: 5
Results:
Jolie, Angelina
Lopez, Maribel
Laguerre, Tony
Loperena, Martha
Santos, Benjamín
Korg, James
Klinger, Heidi

Categoria: INTERMEDIO	CUB Programming Contest Problem set
Author: USACO	May 29,1999

Super Roman Numerals [Kolstad, 1997]

You've heard the story of Romans like Midas who had the `Golden Touch'.Midas was no fool and sold his gold for lots of money. The traditional Roman numerals proved inconvenient for expressing the value of his fortune, whichranged into the millions. He invented Super Roman Numerals.

Super Roman Numerals follow the traditional rules for Roman numerals but have many more single-character values. Consider the traditional Roman numeral values, shown here with the single letter and the decimal number it represents:

I 1	**L** 50	**M** 1000
V 5	**C** 100	
X 10	**D** 500	

As many as three of the same marks that represent 10^n may be placed consecutively:

III is 3
CCC is 300

Marks that are 5 * 10^n are never used consecutively. Generally (with the exception of the next rule), marks are connected together and written in descending order:

CCLXIII = 100+100+50+10+1+1+1 = 263

Sometimes, a mark that represents 10^n is placed before a mark of one of the two next higher values (**I** before **V** or **X**; **X** before **L** or **C**; etc.). In this case, the value of the smaller mark is SUBTRACTED from the mark it precedes:

IV = 4
IX = 9
XL= 40

But compound marks like **XD**, **IC**, and **XM** are not legal, since the smaller mark is too much smaller than the larger one. For **XD** (wrong for 490), one would use **CDXC**; for IC (wrong for 99), one would use **XCIX**; for **XM** (wrong for 990), one would use **CMXC**.

Regrettably, in standard Roman numerals, numbers like 10,000 are represented as **MMMMMMMMMM**. In Super Roman Numerals, the table of marks is extended:

I 1	**L** 50	**M** 1,000	**R** 50,000	**U** 1,000,000	**N** 50,000,000
V 5	**C** 100	**P** 5,000	**S** 100,000	**B** 5,000,000	**Y** 100,000,000
X **10**	D **500**	Q **10,000**	T **500,000**	W **10,000,000**	Z **500,000,000**

Numbers greater than 100 million are now easily expressible. Write a program that reads non-negative decimal numbers (one per line) from the file **INPUT.DAT** and prints the Super Roman Numeral equivalent. It is promised that the input data will require an answer that is representable using the given rules. Stop your program when the input number is a 0.

SOLUCIONES

Super Roman Numerals [Kolstad, 1997]

SAMPLE INPUT (file INPUT.DAT):

```
18
1997
12345678
0
```

SAMPLE OUTPUT:

```
18 XVIII
1997 MCMXCVII
12345678 WUUSSSQRPDCLXXVIII
```

TEST DATA SET #1:

```
18
1998
87654321
99999999
11111
0
```

-

Categoria: INTERMEDIO	CUB Programming Contest Problem set
Author: USACO	May 29,1999

FACTORIALS

The factorial of an integer n, written n!, is the product of all the integers from 1 through inclusive. The factorial quickly becomes very large: 13! is too large to store in a 32-bit integer on most computers, and 70! is too large for most floating-point variables. Your task is to find the rightmost non-zero digit of n!. For example, 5! = 1 * 2 * 3 * 4 * 5 = 120, so the rightmost non-zero digit of 5! is 2. Also, 7! = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040, so the rightmost non-zero digit of 7! is 4.

Input:

An integer n, between 1 and 1000 inclusive.

Output:

The rightmost non-zero digit of n!

TEST CASE 1

Input:
5

Output:
2

-

TEST CASE 2
Input:
789

Output:
4

TEST CASE 3
Input:
1000

Output:
2

This program may be tested by entering in input from the keyboard.

Categoria: INTERMEDIO	CUB Programming Contest Problem set
Author: USACO	May 29,1999

PRIME PALINDROMES

The number 151 is a prime palindrome because it is both a prime number and a palindrome (it is the same number when read forward as backward). Write a program that finds all prime palindromes between two numbers a and b. You may assume that a and b are between 1 and 32,000.

Test your program with a,b = 1, 1000 and a,b =1000, 32000

Test Case

a,b = 1,1000

PRIME PALINDROMES BETWEEN 1 AND 1000

2	3	5	7	11
101	131	151	181	191
313	353	373	383	727
757	787	797	919	929

Test Case 2

a,b = 10000, 32000
-

PRIME PALINDROMES BETWEEN 1000 AND 32000

10301	10501	10601	11311	11411	12421	12721	12821	13331	13831	13931
14341	14741	15451	15551	16061	16361	16561	16661	17471	17971	18181

Universidad de Puerto Rico
Bayamón, Puerto Rico

Competencias de Programación 1999

Principiante

Categoría: Principiante	CUB Programming Contest Problem set
Autor: Prof. Juan Manuel Solá Sloan	May 29, 1999

Validación de Tarjetas de Crédito

Todas las tarjetas de crédito tienen un sistema de verificación de números. Esto se hace para verificar la autenticidad del número de la tarjeta. Cada número tiene un significado. La mayoría de las veces los primeros números agrupados (que suelen ser de cuatro) significan el tipo de tarjeta. Los demás números significan la oficina del representante de servicio y los números que identifican al cliente. Los últimos números se utilizan para verificar si el número de tarjeta de crédito es correcto. A estos se le conocen como los *check digits*.

Construya un programa que dada la siguiente fórmula valide los números de tarjeta de crédito.

1. Los números pares se acumulan.
2. Cada número impar se multiplica por 2 y se acumula.
3. Los *check digits* se calculan buscando el residuo de lo acumulado de números pares entre el acumulado de números impares.

Ejemplo de *input*:

```
8903 9854 9894 8900 48

2943 5564 6864 7910 37

1141 5564 1864 1000 11
```

Ejemplo de output:

```
8903 9854 9894 8900 48  <- Válido

2943 5564 6864 7910 37  <- Válido

1141 5564 1864 1000 11  <- Número de tarjeta erroneo
```

Nota: Utilize archivos para el input y output (ccard.in, ccard.out).

Categoría: PRINCIPIANTE	CUB Programming Contest Problem set
Author: Prof. Antonio Huertas	May 29, 1999

CÁLCULO DE FECHAS

En muchas aplicaciones es necesario determiner fechas anteriores o posteriores a una fecha dada (ej.: para determinar el vencimiento de una tarjeta de crédito). Escriba un programa que reciba una fecha (en formato **MM/DD/AAAA**, donde **MM** es el mes, **DD** es el día y **AAAA** es el año), la valide y determine la fecha del día anterior y del próximo día. Recuerde tomar en consideración que no todos los años son bisiestos y que no todos los meses tienen la misma cantidad de días.

Mes	**Cantidad de Días**	**Mes**	**Cantidad de Días**
1. Enero	31	7. Julio	31
2. Febrero	28 ó 29	8. Agosto	31
3. Marzo	31	9. Septiembre	30
4. Abril	30	10. Octubre	31
5. Mayo	31	11. Noviembre	30
6. Junio	30	12. Diciembre	31

CASOS DE PRUEBA

Input:

05/28/1999
11/35/1999
01/01/1999
02/28/1999
02/29/1999

Output:

Anterior a 05/28/1999 es 05/27/1999 y posterior es 05/29/1999
La fecha entrada es incorrecta. El día debe ser un número entre 1 y 30 para este mes.
Anterior a 01/01/1999 es 12/31/1999 y posterior es 01/02/1999
Anterior a 02/28/1999 es 02/27/1999 y posterior es 03/01/1999
La fecha entrada es incorrecta. 1999no es año bisiesto.

Nota: **UTILICE DE ENTRADA EL ARCHIVO FECHAS.IN**

Hint: Los años bisiestos son divisibles entre 4. Ej. 1992 / 4 = 498

Categoría: PRINCIPIANTE	CUB Programming Contest
	Problem set
Author: Prof. Antonio Huertas	May, 29, 1999

CONVERSION DE NUMEROS HEXADECIMALES

El sistema hexadecimal es utilizado como una alternativa al sistema binario binario para obtener una representación de las cantidades almacenadas en la memoria de una computadora. Este sistema posee dieciseís (16) dígitos. Se puede establecer la siguiente correlación entre los dígitos hexadecimales y cantidades expresadas en el sistema decimal.

Digito Hexadecimal	Equivalente Decimal	Digito Haxadecimal	Equialente Decimal
1	1	9	9
2	2	A	10
3	3	B	11
4	4	C	12
5	5	D	13
6	6	E	14
7	7	F	15
8	8		

Todo número representado en sistema decimal se puede expresar como una suma de potencias de diez (10). Por ejemplo, el número 452 puede ser expresado como (4 * 102) + (5 * 101) + (2 * 100). De igual forma, un número representado en sistema hexadecimal puede ser expresado como la suma de potencias de dieciseís (16). Por ejemplo, el número AD8 puede ser expresado como (A * 162) + (D * 160) = (10 * 162) + (13 * 161) + (8 * 16 0) =2560 + 208 + 8 = 2776

Usted desarrollará un programa que solicite un número hexadecimal con un máximo de ocho (8) dígitos, lo valide y que determine su equivalente decimal.

Nota: Si el número entrado no es válido, se mostratá un mensaje de error y se indicará el primer dígito inválido.

EJEMPLO

Input:
AD8
D7G4
109
FF3

Output:
El equivalente decimal de AD8 es 2776
D7G4 no es una representación hexadecimal. Dígito no válido: G
El equivalente decimal de 109 es 265
El equivalente decimal de FF3 es 4083

Nota: Utilize de input el archivo de nombre HEX.IN

Categoría: Principiante	CUB Programming Contest Problem set
Author: Prof. Juan Solá Sloan	May, 29, 1999

Y2K Software Solution

Date Windowing

El año 2000 encara varios problemas en las computadoras. Uno de ellos en el área de la programación de aplicaciones. Algunas aplicaciones creadas en décadas pasadas utilizan para representar el año en una fecha los últimos dos dígitos del año. Por ejemplo: para el año 1967 se escribe 67. Para restar 1988 de 1985 se representaba así 88-85=3. Como se obviaban los los primeros dígitos del año, ya que no cambiarían hasta el 2000, muchos archivos no guardan el año en cuatro posiciones. Este estilo de programación obsoleta 01-99 = -98 cuando debería ser 2.

Ingéniese un algoritmo de programación para restar fechas entre siglos. Existe una técnica que puede utilizar que se conoce como Date Window. Esta crea una ventana virtual entre siglos cada 99 años. La misma puede ser movida cada 50 años. Por ejemplo: Si entro que mi ventana comienza en el año 1900 mi ventana comienza en 1921 y termina en el año 2000. Si escribo 1920 mi ventana comienza en 1921 y terminará en el 2020. Las fechas mayores del 2020 se tratan como si fueran mayores del 1920. El sistema solo ve una ventana de fechas. Si entro una fecha menor de 1921 el sistema creerá que es menor o igual al 2020.

Input:

```
1900 69 68
1850 25 88
2220 01 99
1800 90 81
```

Output:

```
Ventana : 1901 al 2000, 1969-1968=1
Ventana:  1851 al 1950, 1925-1888=37
Ventana:  2221 al 2320, 2301-2299=2
Ventana:  1801 al 1900, 1890-1881=9
```

Nota: Utilize archivos de input y output para este problema (llamarlos: Y2K. IN y Y2K. OUT)

PROGRAM LISTING

Una cualidad que parece disminuir con el uso de los lenguajes de programaci6n en ambiente de PC es la impresion en papel de un "source program" que pueda listar el codigo con enumeraci6n por linea y encabezamiento en cada prigina. Esta opci6n se usa mucho en ambientes de "Mainframe" con listados de programas largos que se imprimen con la opci6n "LISTING" para poder examinar detenidamente el programa, sus variables e incluso los mensajes de errores.

La opcion de "LISTING" no solo genera un listado con encabezamiento por pagina y enumeraci6n por linea, sino que tambien lista los errores de sintaxis si es que existe alguno yen que linea se encuentra. Tambien puede generar un listado de variables, en donde se definen, en donde se modifican yen donde se utilizan. Esto era de gran ayuda a los programadores de ambiente "Mainframe" y tambien considero que puede ser de ayuda a los programadores de ambiente PC.

PROBLEMA

Haga un programa que lea un programa de Pascal de entrada y genere de salida un encabezamiento por pagina y enumere cada linea de codigo que tenga el programa.

PUNTOS IMPORTANTES

Tenga en mente lo siguiente

1. El programa debe imprimir hasta 55 lineas por prigina.

2. En la primera parte del encabezamiento se utilizarfi el nombre del programa puesto despues de PROGRAM con la extension PAS.

3. En la segunda parte se incluirfi la fecha y hora de la computadora y el dia de la semana.

4. Finalmente en la tercera parte se va enumerando las páginas.

5. Después de enumerar en cada línea, se pone el caracter "colon" (:)y se deja por lo menos un espacio en blanco.

EJEMPLO DE CORRIDA

Utilizando de ejemplo el siguiente c6digo:

```
PROGRAM newton (input, output);

CONST
        Epsilon = le-6;

VAR
     Number, root, sqroot: real;

BEGIN
     REPEAT
          writeIn;
          writeCEnter new number (0 to quit): ');
          read(number);

          IF number = 0 THEN BEGIN
               Writein(number: 12:6, 0.0:12:6);
          END
          ELSE IF number < 0 THEN BEGIN
               WriteIn('*** ERROR: number < 0');
          END
          ELSE BEGIN
               Sqroot := sqrt(number);
               writein(number: 12:6, sqroot: 12: 6);
               writeln;

               root := 1;
               REPEAT
                    root := (number/root + root)/2;
                    writein(root: 24: 6,
                         100*abs(root - sqroot)/sqroot: 12: 2, ,
                         %,)
               UNTIL abs(number/sqr(root) - 1) < epsilon;
          END
     UNTIL number = 0

END.
```

Se debe generar el siguiente listado:

newton.pas Sat May 29, 1999 10:13:25 Page: 1

```
 1: PROGRAM newton (input, output);
 2:
 3: CONST
 4:            Epsilon = le-6;
 5:
 6:  VAR
 7:            Number, root, sqroot : real;
 8:
 9:  BEGIN
10:       REPEAT
11:             writeln;
12:             write('Enter new number (0 to quit): ');
13:             read(number);
14:
15:             IF number =0 THEN BEGIN
16:                    Writeln(number:12:6, 0.0:12:6);
17:             END
18:             ELSE IF number <0 THEN BEGIN
19:                  Writeln('***ERROR:  number <0');
20:             END
21:             ELSE BEGIN
22:                  Sqroot :=sqrt(number);
23:                  writeln(number:12:6, sqroot: 12:6):
24:                  writeln;
25:
26:                  root := 1;
27:                  REPEAT
28:                      root := (number/root + root)/2;
29:                      writeln(root:24:6,
30:                              100*abs(root - sqroot)/sqroot:12:2,
31:                              '%')
32:                  UNTIL abs(number/sqr(root) - 1) <epsilon;
33:              END
34:       UNTIL number = 0
35:  END.
```

Universidad de Puerto Rico
Bayamón, Puerto Rico

Competencias de Programación 1999

Experto

Code Generation

Your employer needs a backend for a translator for a very SIC machine (Simplified Instructional Computer, apologies to Leiand Beck). Input to the translator will be arithmetic expressions in postfix form and the output will be assembly language code.

The target machine has a single register and the following instructions, where the operand is either an identifier or a storage location.

L	**ST**
A	load the operand into the register
S	add the operand to the contents of the register subtract the operand from the contents of the
M	register multiply the contents of the register by the operand divide the contents of the register by
D	the operand negate the contents of the register store the contents of the register in the operand
N	location

An arithmetic operation replaces the contents of the register with the expression result. Temporary storage locations are allocated by the assembler for an operand of the form OSnO where n is a single digit.

Input and Output

The input file consists of several legitimate postfix expressions, each on a separate line. Expression operands are single letters and operators are the normal arithmetic operators (+, -, *,/) and unary negation (@). Output must be assembly language code that meets the following requirements:

1. One instruction per line with the instruction mnemonic separated from the operand (if any) by one blank.
2. One blank line must separate the assembly code for successive expressions.
3. The original order of the operands must be preserved in the assembly code.
4. Assembly code must be generated for each operator as soon as it is encountered.
5. As few temporaries as possible should be used (given the above restrictions).
6. For each operator in the expression, the minimum number of instructions must be generated (given the above restrictions).

A sample input file and corrresponding correct output are on the reverse of this paper.

Sample input	**Sample output**
AB+CD+EF++GH+++	L A A B ST $1 L C A D ST $2 L E A F A $2 ST $2 L G A H A $2 A $1

AB+CD+-	L A A B ST $1 L C A D N A $1

CONVERSION

Convierta de palabra a número dejándose llevar de la siguiente tabla:

primero = 1	undecimo = 11	Vigesimo primero = 21	ducentesimo = 200
segundo = 2	duodecimo = 12	trigesimo = 30	tricentesimo = 300
tercero = 3	decimotercero 6 decimotercio = 13	trigesimo cuarto = 34	cuadrigentesimo = 400
cuarto = 4	decimocuarto = 14	cuadragesimo = 40	quingentesimo = 500
quinto = 5	decimoquinto = 15	quincuagesimo = 50	sexcentesimo = 600
sexto = 6	decimosexto = 16	sexagesimo = 60	septingentesimo = 700
septimo = 7	decimoseptimo = 17	septuagesimo = 70	octigentesimo = 800
octavo = 8	decimoctavo = 18	octogesimo = 80	nonigentesimo = 900
noveno = 9	decimonoveno 6 decimonono = 19	nonagesimo = 90	milesimo = 1000
decimo = 10	vigesimo = 20	centesimo = 100	

Se tomará 1o siguiente en mente:

1. Se omitirán los acentos en las palabras.
2. Las cifras llegan hasta un máximo de cuatro (4) dígitos.
3. El programa tiene que detectar errores de sintaxis.

Ejemplo de corrida:

Indique palabra (exit para salir): quincuagesimo segundo
q**uincuagesimo seRundo = 52**

Indique palabra (exit para salir): centesimo sexagesimo quinto
centesimo sexagesimo quinto = 165

Indique palabra (exit para salir}: milesimo primero
milesimo primero = 1001

Indique palabra (exit para salir}: centesimo nonagesimo
centesimo nona2esimo = 190

lndique palabra (exit para salir): septima
*****error de sintaxis *****

Indique palabra (exit para salir): exit

CUTB Programming Contest
Problem Set-Expert Division
March 6, 1999

Pascal to Assembler Converter

Input Data Name: PASCAL.TXT
Output File Name: PASCAL.TXT

Define

The **repeat - until** statement in the Pascal language has the following format:

repeat *action 1*
action 2
action 3
.
.
action n
until <condition>

Problem:

Write a program to translate valid **repeat** statements in Pascal to the Assembly language. Your program will read a Pascal **repeat** statement from the file PASCAL.TXT, and will write the equivalent assembly statement into the file ASSEM.TXT.

Sample Run:

Input:

```
repeat ax := ax-1
     bx = bx + 1
     cx = ax - bx
until ax = 0
```

Output:

```
   START            DEC   AX
REPEAT:             INC   BX
                    MOV   CX,0
                    ADD   CX, AX
                    SUB   CX, BX
                    CMP   AX,0
                    JNZ   START REPEAT
```

There will be only simple mathematics equotions like add, subtract, divide and multiply with a maximum of three (3) operators. The compare symbols like >, <, >=, <=, <> and = may also be used in the **until** clause.

You can use the following assembly instructions:

1. JMP - Jump no matter what
2. JZ - Jump if the result was Zero
3. JNZ - Jump if the result was Not Zero
4. JG - Jump if the result was Greater than zero
5. JL - Jump if the result was Less than zero
6. JGE - Jump if the result was Greater or equal than zero
7. JLE - Jump if the result was Less or equal than zero
8. DEC - Decrement (subtract one)
9. INC - Increment (Add one)
10. MOV - Move X,Y (X (''' Y)
11. ADD - Add X,Y (X (- X +Y)
12. SUB - Subtract X,Y (X (- X -Y)
13. MUL- Multiply X,Y (X (- X *Y)
14. DIV - Divide X,Y (X (-- X / Y)
15. CMP - Compare

Universidad de Puerto Rico
Bayamón, Puerto Rico

Competencias de Programación 1997

Experto

Fecha: Mayo/03/1997	**Nombre de la competencia**: CUTB Programmig Contest
Categoría: Experto	**Universidad**: _____
Autor: _____	**Tipo de competencia**: _____
Problema #: _____	**Algoritmos**: _____

BINARY CALCULATOR

Source File Name: XXXXXXXX.XXX

Input File Name: XXXXXXXX.XXX

Output File Name: XXXXXXXX.XXX

Problem 1:

You are to write a program that can evaluate simple expressions using binary numbers. The expression can allow the following tokens:

Binary numbers: Optional sign, the characters **1** and **0**
Operators: +,-,*, or /
Parenthesis: **(** or **)** to specify precedence grouping.

You must assume:

1. The expression should be evaluated right to left (except where parenthesis grouping is found).
2. All input expressions are syntactically correct.
3. Each token is separated by one blank space.
4. All expressions will be less than 50 characters in length.
5. All operations will produce integer results only (ignore decimal positions).

When an input expression is read, you are to print that expression and its correct value.
YOU CAN NOT CONVERT FROM BINARY TO DECIMAL TO SOLVE THIS PROBLEM.

Examples:

ENTER EXPRESSION>**1101□+□1111**
1101□+□ 1111 = 11100

ENTER EXPRESSION>**1101□*□1111**
1101□*□1111 = 11000011

ENTER EXPRESSION>**101□/□10**
101□/□10 = 10 R 1

ENTER EXPRESSION>**0**
<EXIT>

Note that □ is equivalent to a space.

Fecha: Mayo/03/1997	**Nombre de la competencia**: CUTB Programming Contest
Categoría: Experto	**Universidad**: _____
Autor: _____	**Tipo de competencia**: _____
Problema #: _____	**Algoritmos**: _____

Directory Listing Command Simulator

Source File Name: XXXXXXXX.XXX

Input File Name: XXXXXXXX.XXX

Output File Name: XXXXXXXX.XXX

Problem 2:

Develop a program that simulates a generic **DIR** command. The input will be read from an input file named **DIR.TXT**. The characters allowed in that file will be:

1. A dot **(.)** to separate the file name and the extension (optional).
2. Characters from **A** to **Z** (Upper and Lowercase will be allowed).
3. Numbers from **0** to **9**.

The prompt command will allow the following characters:

1. A dot **(.)** (optional)
2. An asterisk **(*)** to substitute one or more characters.
3. A question mark **(?)** to substitute only one character.

The following are examples of valid commands.

DIR*a.EXE, DIR AB*.COM, DIR A?B?C?.DAT, DIR ABC?H*.*, DIR HELP, DIR *.*, DIR ????????.???, DIR *A?B*.*

The rules for file names will be the same used for MS-DOS. At the end of the program will display the totals of files displayed on screen and the total files read on the file. Remember, the program must be case sensitive.

Sample output:

COMMAND>DIR*.EXE
ABC.EXE
FINISH.EXE
PROGRAM.EXE
TEST.EXE
Total files read (25) Total files selected (4).

COMMAND>DIR *H.*
FINISH.EXE
ASH.TXT
Total files read (25) Total files selected (2).
COMMAND>DIR A?B?C*.*
No files found
Total files read (25) Total files selected (0).

COMMAND>DIR A?C*.*
ABC.EXE
AXZTOT.BAT
Total files read (25) Total files selected (2).

Problem 3: GOLDBACH CONJECTURE

Source File Name: ED3.XXX
Input File Name: ED3.DAT
Output File Name: ED3.OUT

Define:

The Goldbach conjecture states that any positive even number greater than 4 can be expressed as the sum of two prime numbers. This conjecture has never been completely proven, but it has been demonstrated by computer to be true for a wide range of even numbers.

Problem:

Given an even number greater than 4, find two prime numbers which sum to it. For purposes of this problem, 1 is not considered a prime number.

Input:

Input for this problem consists of a list of even numbers greater than 4, one per line. The numbers will be fewer than 10 digits in length.

Output:

Each line of the program output consists of exactly three entities: the original input number and the two primes which sum to that number.

Sample Data:

8
10

Sample Output:

Original	Prime	
8	3	5
10	5	5

ENTER AN EVEN NUMBER>0
<EXIT>

EXPERT DIVISION

LONG, LONG DIVISION

Problem: 4

You have been assigned to a team of software-hardware engineers working on super computers development. Your task is to write software for implementing multiple digit division, which is to divide any integer of 40 or fewer digits by any positive divisor less than 100.

Data in the input files comes in pairs, with the first line containing the dividend and the second line containing the divisor. Your program is to accept only correct dividends and divisors. Thus, if either the dividend or its divisor contains any non-digit, i.e., a character not in [0..9], or the divisor is greater than 99 you are to print an error message, as shown in the sample out shown below.

If you read in two valid values, you are to compute the quotient and remainder and output the results as shown on the sample output shown below. You are to use normal end-of-file methods to terminate your reads from the input data file. Always skip a line, as shown below in the example data, between the dividend/divisor pairs.

Sample Data:

```
9
3
53
7
```

Sample output:

```
ENTER FIRST NUMBER> 9

ENTER SECOND NUMBER> 3

Dividend is 9

    Divisor is 3
    Quotient is 3
    Remainder is 0

ENTER FIRST NUMBER> 53
ENTER SECOND NUMBER> 7
    Dividend is 53
    Divisor is 7
    Quotient is 7
    Remainder is 4

ENTER FIRST NUMBER> 0
```

Universidad de Puerto Rico
Bayamón, Puerto Rico

Competencias de Programación 1997

Intermedio

INTERMEDIATE DIVISION

DEALING A DECK OF CARDS

Problem 1:

Write a program that, given a list of the integers 1 to 52 in any order whatsoever, simulates the dealing of a deck of cards. You must assign some correspondence between the numbers from to 1 to 52 and the cards in a standard deck of 52 cards. One way to do this is to let the first 13 cards be hearts, the next 13 be diamonds, the next 13 be clubs, and the last 13 be spades. Within each suit of 13 cards, the first 10 cards represent the ace through the ten, and the remaining three cards represent the jack, the queen, and the king of that suit, respectively.

On the basis of the scheme presented above, the number 1 corresponds to the ace of hearts, the number 2 corresponds to the two of hearts, the number 26 correspond to the king of diamonds, the number 39 corresponds to the king of clubs, and the number 52 corresponds to the king of spades.

You musts also distribute the dealed cards to 4 players (4 per player) from the top and giving one card to each player one at a time.

<u>YOU MUST USE THE RAMDOM FUNCTION TO SOLVE THIS PROBLEM!!</u>

Sample Output:

<UNDEALING>

1. ace of hearts
2. two of hearts
3. three of hearts
4. four of hearts

.
.

52. King of spades

<**DEALING>**

1. Ace of Clubs	14. Nine of Diamonds	27.Five of Hearts	40. Five of Clubs
2. Four of Hearts	15. Ten of Spades	28.Deuce of Diamonds	41. Six of Hearts
3.Nine of Spades	16.King of Clubs	29.Queen of Spades	42.Nine of Hearts
4. Ace of Hearts	17.King of Spades	30.King of Hearts	43.Ten of Hearts
5. Eight of Diamonds	18.Seven of Clubs	31. Four of Spades	44.Deuce of Clubs
6. Ace of Diamonds	19.Six of Clubs	32.Six of Diamonds	45.Three of Hearts
7. Jack of Hearts	.	.	.
8. Seven of Hearts .	.	.	
.	.	.	.
13. Queen of Clubs	26.Deuce of Spades	.39.Jack of Spades	52. Five of Diamonds

PLAYER 1	**PLAYER 2**	**PLAYER 3**	**PLAYER 4**
1. Ace of Clubs	1.Four of Hearts	1. Nine of Spades	1. Ace of Hearts
2. Eight of Diamonds	2. Ace of Diamonds	2. Jack of Hearts	2. Seven of Hearts
3. .	3. .	3. .	3. .
4. Queen of Clubs	4. Nine of Diamonds	4. Ten of Spades	4. King of Clubs

INTERMEDIATE DIVISION

FRACTIONS TO DECIMALS

Problem 2:

Write a program that will accept a fraction of the form N/D, where **N** is the numerator and **D** is the denominator, that prints out the decimal representation. If the decimal representation has a repeating sequence of digits, it should be indicated by enclosing it in brackets. For example, 1/3 =.33333333…is denoted as .(3)0, and 41/333+.123123123…is denoted as. (123).

Typical conversions are:

1/3 = .(3)
22/5 = 4.4
1/7 = .(142857)
3/8 = .375
45/56 = .803 (9571428)

Test your program with the fractions above and the fraction 11/59.

Sample Run

ENTER N> 1
ENTER D>7

1/7=.(142857)

ENTER N>0
<EXIT>

INTERMEDIATE DIVISION

EIGHT QUEEN WITH A TWIST

Problem 3:

Eight queens can be arranged on a chess board so that no Queen is under attack from any of the others; in other words, so that no row or column or diagonal contains more than one queen. Write a program that will place the position of 8 queens on a chess board and ensures that no Queens are under attack. Your program should first draw the chess board displaying the **X Y** matrix with Q's representing the Queen position. The program must also search and display all the solutions for this problem. The answers should show the correct solution.

Note: No hard-wired solution are allowed!.

Sample Output:

	1	2	3	4	5	6	7	8
1	Q							
2			Q					
3								Q
4						Q		
5				Q				
6		Q						
7					Q			
8							Q	

PRESS ENTER FOR NEXT SOLUTION>

INTERMEDIATE DIVISION

PUZZLE

Problem 4:

You are to write a program which will scan a 10 x 10 character grid, and find the occurrence of a particular word in the grid. The word can be found horizontally, vertically, diagonally **ON ANY DIRECTION**. The input filename will be PUZZLE.DAT.

For instance, given a grid: (4x4)

A	B	D	X
K	O	H	P
H	W	U	P
P	L	U	P

Sample Output:

ENTER WORD (4 characters long)> BOWL
BOWL is in positions, (2,1), (2,2), (2,3), (2, 4).

ENTER WORD(4 characters long)>PULP
PULP is in positions, (44), (34), (2, 4), (1,4).

ENTER WORD (4 characters long)> PULL
The word PULL is not in the grid.

ENTER WORD (4 characters long)>0
<EXIT>

Universidad de Puerto Rico
Bayamón, Puerto Rico

Competencias de Programación 1997

Principiantes

BEGINNERS DIVISION

EXPONENTIATION

Problem 1:

Write a program that implemems exponentiation. The following rules must be observed.

1. For the following expression: N^e, where N and e are integers.
   When e > 0, N^e = (N * .... N) e times
   Whene=0, N^e =1
   When e < 0, N^e = 1/(N * ..... N) e times

2. <u>DO NOT USE ANY MATHEMATICAL BUILT-IN FUNCTION</u> to solve this problem, other than +, -, * and/.

Sample Data:

ENTER NUMBER>5
ENTER EXPONENT>3

Sample Output:

RESULT = 125

BEGINNERS DIVISION

DEALING A DECK OF CARDS

Problem 2:

Write a program that e-iven a list of the integers 1 to 52 in any order whatsoever, simulates the dealing of a d, of cards. You must assign some correspondence between the numbers from 1 to 52 and the cards in a standard deck of 52 cards. One way to do this is to let the first 13 cards be hearts, the next 13 be diamonds, the n,-xt 13 be clubs, and the last 13 be spades. Within each suit of 13 cards, the first 10 cards represent the ace through the ten, and the remaining three cards represent the jack, the queen, and the king of that suit, respectively.

On the basis of the scheme presented above, the number 1 corresponds to the ace of hearts, the number 2 corresponds to the two of hearts, the number l0 corresponds to the ten of hearts, the number 13 correspond to the king of hearts, the number 26 correspond to the king of diamonds, the number 39 corresponds to the king of clubs, and the number 52 corresponds to the king of spades.

<u>YOU MUST USE THE RANDOM FUNCTION TO SOLVE THIS PROBLEM!!</u>

Sample Output:

<UNDEALING>
1. ace of hearts
2. two of hearts
3. three of hearts
4. four of hearts
.
.
.
52. King of spades

<DEALING>
1. Ace of Clubs
2. Four of Hearts
3. Nine of Spades
4. Ace of Hearts
5. Eight of Diamonds
6. Ace of Diamonds
.
.
.
52. Queen of Clubs

PRESS ENTER TO EXIT>

BEGINNERS DIVISION

SUBTRACTING BIG NUMBERS

Problem 3:

Write a program that subtract two numbers up to 32 characters long. Both numbers must be included in the same line and they will be separated by a blank space. Also assume both numbers are integers and positives.

Example Output:

ENTER NUMBERS> 400 □321
400 - 321 = 79

ENTER NUMBERS> 1000 □2001
1000- 2001 =-1001

ENTER NUMBERS> 0
<EXIT>

BEGINNERS DIVISION

FACE OF THE CLOCK

Problem 4:

Write a program that inputs a time such as 11:35 (35 minutes after the hour 11), and then draws the face of a clock with the hour and minute hands positioned to indicate the time. The program needs to validate input time. You have free format to design the clock (in CHARACTER environment only), but the time must be legible to the user. You can use the full screen to display time if you want.

Example Output:

ENTER TIME> 11:35

```
     / 11        12        1 \
   /        *              \
 /10          *           2 \
 /               *               \
 9                            3
 \               ,               /
 \8           *             4/
 \          *                /
 \7       *                  5/
 \               6           /
```

PRESS ENTER TO EXIT>

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

PROBLEMAS PARA ELIMINATORIAS

CATEGORIA: INTERMEDIOS

PROBLEMA # 1

Escriba un programa que solicite al usuario una palabra de 5 letras. Luego tiene que tirar por pantalla todas las posibles combinaciones (permutaciones) de esa palabra. El programa tiene que validar la entrada de datos de modo que solo permita aceptar letras y el largo tiene que ser igual a cinco. (Ver anexo.)

PROBLEMA # 2

Escriba un programa que solicite al usuario el número máximo de una lista. Luego procede a tirar por pantalla comenzando en uno hasta el número dado, las siguientes representaciones:

Decimal, Octal, Hexadecimal y Binario.

Ejemplo:

Entre un número (no mayor de 200): 8

Decimal	Octal	Hexadecimal	Binario
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000

El programa debe validar la entrada de datos y cuando la pantalla se llene, debe pausar para que el usuario pueda ver los datos e indicar cuando desea continuar.

Universidad de Puerto Rico
Bayamón, Puerto Rico

Competencias de Programación 1996

Experto

EXPERT DIVISION

CABRA COMPILER

Problem 3:

Input File Name:
Output File Name:
CABRA. SRC CABRA. OUT

Definition:

A string is any combination of blank, letters, digits and special characters. Words are groups of characters separated by blanks.

Problem:

Develop a **CABRA** language parser/compiler. The **CABRA** (Compiler And Basic Reference Assembler) language has the following features:

1. Can handle up to **20** variables, all of which are global.
2. Variables can be up to **6** characters long.
3. Only strings (up to **40**) can be assigned to variables.
4. A variable can store strings up to **40** characters.

A **CABRA** programmer can use:

1. A **COUNT** function, which will count, the amount of characters whitin a string (one argument).
2. A **PRINT** function, which will print, the value for a variable (one argument).
3. A **BEGIN** symbol, which is used to mark the beginning of a program.
4. An **END** symbol, which is used to mark the end of a program.
5. An assignment symbol (=), which is used to assign strings to variables.

All functions are reserved symbols. The **CABRA** compiler should print, an error if.'

1. Any function name is used as a symbol.
2. Any unassigned symbol is used in a function.
3. Any undefined function is used.
4. Any syntax error. **(DO NOT DESCRIBE THE I~RROR!!!)**

Additional information:

1. No nested functions are allowed.
2. No need to check for case sensitivity.
3. **THE COMPILER MUST READ UP TO 5 PROGRAMS AS** DATA.

A sample CABRA program is:

```
BEGIN
A1 = "VACA";
A2 = "CABALLO";
A3 = "GALLINA";
C1 = COUNT (A1)',
C2 = COUNT (A2);
C3 = COUNT (A3)
PRINT (A1);
PRINT (C1);
PRINT (A2);
PRINT (C2);
PRINT (A3);
PRINT (C3);
END
```

```
BEGINNING
CABRA- "YES";
PRINT (CABRA),
END
```

And its output will be:

Program # 1:
VACA
4
CABALLO
7
GALLINA
7

Program #2:
Syntax Error at Line **1**

Universidad de Puerto Rico
Bayamón, Puerto Rico

Competencias de Programación 1996

Principiantes

Fecha: April 27 1996	**Nombre de la competencia**: Programming Contest
Categoría: Begenners Divesion	**Universidad**: U P R B
Autor:	**Tipo de competencia**:
Problema #: 01	**Algoritmos**:

MORSE CODE

Source File Name: MORSE. IN

Input File Name: MORSE. COD

MORSE. DEC

Definition:

Perhaps the most famous of all coding schemes is the Morse code, developed by Samuel Morse in 1873 for use with the telegraph system. The Morse code assigns a series of dots and dashes to each letter of the alphabet, each digit, and a few special characters (such as the period, comma, colon and semicolon). In sound-oriented systems, the dot represents a short sound and the dash represents a long sound. Other representations of dots and dashes are used with light-oriented systems and signal flag systems.

Separation between words is indicated by a space, or, quite simply, the absence of a dot or dash. In a sound-oriented system, a space is indicated by a short period of time during which no sound is transmitted. The international version of' the Morse code is the following table:

MORSE CODE

A	**.-**	**J**	**.---**	**S**	**...**
B	**-...**	**K**	**-.-**	**T**	**-**
C	**-.-.**	**L**	**.-..**	**U**	**..-**
D	**-..**	**M**	**--**	**V**	**...-**
E	**.**	**N**	**-.**	**W**	**.--**
F	**..-.**	**O**	**---**	**X**	**-..-**
G	**--.**	**P**	**.--.**	**Y**	**-.--**
H	**...**	**Q**	**--.-**	**Z**	**--..**
I	**..**	**R**	**.-.**		

Problem:

Write a program that reads several lines of text (**MORSE.IN**) and then create a new one using the Morse code (**MORSE.COD**). Read the new file and decodeit to text again (**MORSE.DEC**). Assume **4** characters of Morse code per character.

Fecha: April 27 1996	**Nombre de la competencia**: CUTB Programmuing Contest
Categoría: Begenners Divesion	**Universidad**: U P R B
Autor:	**Tipo de competencia**:
Problema #: 2	**Algoritmos**:

MASTERMIND

Problem 2
Definition:

Mastermind is a logic-guessing game. The object is to guess four Code Peg colors in a row. The game must be played by two persons. The first one chooses any combination of four Code Peg colors. The second one tries to guess that combination selecting any four combinations of the available Code Peg colors. The first player compares the colors and notifies the other player this way:

1. A Black Key Peg indicates a Code Peg of the right color and in the right position.
2. A White Key Peg indicates a Code Peg of the fight color but in the wrong position.
3. No Key Peg indicates a wrong color that does not appear in the secret code.

The second player has five more opportunities to guess the pattern colors using as a guide the Key pegs the First player indicate.

Problem:

Develop a program with the following requirements:

1. Select 4 colors (random). The available colors are:
   a. **RD** = Red
   b. **BU** = Blue
   c. **YW** = Yellow
   d. **GN** = Green
   e. **OR** = Orange
   f. **BN** = Brown
2. Input Four valid Colors from the user.
3. Examine the colors entered and respond to the user.Valid answer are:
   a. **WH** = White
   b. **BK** = Black
   c. **NO** = None
4. If the user does not guess in six attempts, end the program.
5. The program must show the user the pattern of colors (for verification purposes)

Sample Data

SELECTED COLORS = RD, YW, OR, BN

ENTER COLORS # **1**> **RD, BU, BN, YW**
MATCH: **BK, WH, WH, NO**

ENTER COLORS # **2**> **RD, OR, BN, YW**
MATCH: **BK, WH, WH, WH**

ENTER COLORS # **3**> **RD, YW, BN, OR**
MATCH: **BK, BK, WH, WH**

ENTER COLORS # **4**> **RD, YW, OR, BN**
MATCH: **BK, BK, BK, BK**

YOU WIN !!!!!

(SECOND GAME)

SELECTED COLORS = RD, YW, RD, YW

ENTER COLORS # **1**> **RD, BU, BN, YW**
MATCH: **BK, BK, NO, NO**

ENTER COLORS # **2**> **YW, BU, BN, RD**
MATCH: **WH, WH, NO, NO**

ENTER COLORS # **3**> **RD, BU, BN, YW**
MATCH: **BK, BK, NO, NO**

ENTER COLORS # **4**> **RD, OR, OR, YW**
MATCH: **BK, BK, NO, NO**

ENTER COLORS # **5**> **RD, GN, GN, YW**
MATCH: **BK, BK, NO, NO**

ENTER COLORS # **6**> **RD, RD, RD, YW**
MATCH: **BK, BK, BK, NO**
YOU LOSE!!!!!

Fecha: April 27 1996	**Nombre de la competencia**: Programming Contest
Categoría: BEGINNERS DIVISION	**Universidad**: U.P.R.B.
Autor:	**Tipo de competencia**:
Problema #:03	**Algoritmos**:

TEXT COUNT

Problem 3:

Input File Name: TEXTBEG.IN
Output File Name: TEXTBEG.OUT

Definition:

The availability of computers with string manipulation capabilities has resulted in some rather interesting approaches to analyzing the writings of great authors, Much attention has been focused in recent years on the issue of whether or not William Shakespeare ever lived. Some scholars believe that there is substantial evidence that indicates that Christopher Marlowe actually penned the masterpieces normally attributed to Shakespeare. Researchers have used computers in efforts to find similarities in the writings of these two authors.

Problem:

Write a program that reads several lines of text and then displays the following information:

1. A table indicating the number of occurrences of each letter of the alphabet.
2. A table indicating the number of one-letter words, two-letter words, three-letter words,etc. appearing in the text (up to ten letters per word).

Uppercase and Lowercase are ignored.

Sample Output:

LETTERS TOTALS

A	5	J	2	S	3
B	2	K	0	T	4
C	0	L	1	U	4
D	1	M	1	V	1
E	8	N	I	W	0
F	2	O	3	X	0
G	1	P	1	Y	0
H	3	Q	0	Z	1
I	4	R	2		

WORD TOTALS

1-LETTERS WORD	**6**	**6-LETTERS WORD**	**1**
2-LETTERS WORD	**3**	**7-LETTERS WORD**	**2**
3-LETTERS WORD	**5**	**8-LETTERS WORD**	**0**
4-LETTERS WORD	**4**	**9-LETTERS WORD**	**0**
5-LETTERS WORD	**6**	**10-LETTERS WORD**	**1**

Fecha: April 27 1996	**Nombre de la competencia**: Programming Contest
Categoría: BEGINNERS DIVISION	**Universidad**: U.P.R.B.
Autor:	**Tipo de competencia**:
Problema #:03	**Algoritmos**:

MEASUREMENT AND UNIT CONVERSION

Problem:

Write a MENU-DRIVEN program that allows the user the following options:

1) Convert measurements from minutes to hours (two decimal places).
2) Convert feet to meters (**1** foot = **0.3048** meter, two decimal places)
3) Convert from degrees Fahrenheit to degrees Celsius (**F** = **1.8 C** + 32).
4) Exit Program.

UNIVERSIDAD DE PUERTO RICO

RECINTO DE MAYAGUEZ

University of Puerto Rico
Mayaguez Campus

ICOM Challenge 2000

Expert Division

Sponsored by

***AEIC* and Lucent Technologies**

Table of Contents

Problem **Page**

Fecha: _04_/03/2000	**Nombre de la competencia**: __ICOM Challenge_
Categoría: __Experto_	**Universidad**: Universidad de Puerto Rico, Mayaguez
Autor: _____	**Tipo de competencia**: _____
Problema #: _____	**Algoritmos**: _____

Input File: huffman.in
Output File: huffman.out
Source File: huffman.xxx

Variable Radix Huffman Encoding

Problem Description

Huffman encoding is a method of developing an optimal encoding of the symbols in a *source alphabet* using symbols from a *target alphabet* when the frequencies of each of the symbols in the source alphabet are known. Optimal means the average length of an encoded message will be minimized. In this problem you are to determine an encoding of the first N uppercase letters (the source alphabet, S_1 through S_N, with frequencies f_1 through f_n) into the first R decimal digits (the target alphabet, T_1 through T_R).

Consider determining the encoding when R=2. Encoding proceeds in several passes. In each pass the two source symbols with the lowest frequencies, say S_1 and S_2, are grouped to form a new "combination letter" whose frequency is the sum of f_1 and f_2. If there is a tie for the lowest or second lowest frequency, the letter occurring earlier in the alphabet is selected. After some number of passes only two letters remain to be combined. The letters combined in each pass are assigned one of the symbols from the target alphabet.

The letter with the lower frequency is assigned the code 0, and the other letter is assigned the code 1. (If each letter in a combined group has the same frequency, then 0 is assigned to the one earliest in the alphabet. For the purpose of comparisons, the value of a "combination letter" is the value of the earliest letter in the combination.) The final code sequence for a source symbol is formed by concatenating the target alphabet symbols assigned as each combination letter using the source symbol is formed.

The target symbols are concatenated in the reverse order that they are assigned so that the first symbol in the final code sequence is the last target symbol assigned to a combination letter.

The two illustrations below demonstrate the process for *R*-2.

Symbol	Frequency
A	5
B	7
C	' 8
D	15

Pass 1: A and B grouped
Pass 2: {A.B} and C grouped
Pass 3: {A,B,C} and D grouped
Resulting codes: A=110, B=1 l 1, C=10, D=10
Avg. Length=(3*5+3*7+2*8+l*15)/35-1.91

Symbol	Frequency
A	7
B	7
C	7
D	7

Pass 1: A and B grouped
Pass 2: C and D grouped
Pass 3: {A,B} and {C,D} grouped
Resulting codes: A=00, B=01, C=10, D=11
Avg. Length=(2*7+2*7+2*7+2*7)/28=2.00

When *R* is larger than 2, *R* symbols are grouped in each pass. Since each pass effectively replaces *R* letters or combination letters by 1 combination letter, and the last pass must combine *R* letters or combination letters, the source alphabet must contain *k*(R-1)+R* letters, for some integer *k*.

Since *N* may not be this large, an appropriate number of fictitious letters with zero frequencies must be included. These fictitious letters are not to be included in the output. In making comparisons, the fictitious letters are later than any of the letters in the alphabet.

Now the basic process of determining the Huffman encoding is the same as for the *R*=2 case. In each pass, the *R* letters with the lowest frequencies are grouped, forming a new combination letter with a frequency equal to the sum of the letters included in the group. The letters that were grouped are assigned the target alphabet symbols 0 through R-1. 0 is assigned to the letter in the combination with the lowest frequency, 1 to the next lowest frequency, and so forth. If several of the letters in the group have the same frequency, the one earliest in the alphabet is assigned the smaller target symbol, and so forth.

The illustration below demonstrates the process for R=.3.

Symbol	Frequency
A	5
B	7
C	8
D	15

Pass 1: ? (fictitious symbol), A and B are grouped
Pass 2: {?,A,B), C and D are grouped
Resulting codes: A=I I, B=12, C=0, D=2
Avg. Length=(2*5+2*7+ 1*8+ 1*15)/35=1.34

Input

The input will contain one or more data sets, one per line. Each data set consists of an integer value for *R* (between 2 and 10), an integer value for *N* (between 2 and 26), and the integer frequencies f_1 through f_N, each of which is between 1 and 999.

The end of data for the entire input is the number 0 for R; it is not considered to be a separate data set.

For each data set, display its number (numbering is sequential starting with 1) and the average target symbol length (rounded to two decimal places) on one line. Then display the *N* letters of the source alphabet and the corresponding Huffman codes, one letter and code per line. The examples below illustrate the required output format.

Sample Input

```
2 5 5 10 20 25 40
2 5 4 2 2 1 1
3 7 20 5 8 5 12 6 9
4 6 10 23 18 25 9 12
0
```

Sample Output

```
Set 1; average length 2.10
    A: 1100
    B: 1101
    C: 111
    D: 10
    E: 0

Set 2; average length 2.20
    A: 11
    B: 00
    C: 01
    D: 100
    E: 101

Set 3; average length 1.69
A: 1
B: 00
C: 20
D: 01
E: 22
F: 02
G: 21
```

Set 4; average length 1.32
A: 32
B: 1
C: 0
D: 2
E: 31
F: 33

Input File Name: sort.in
Output File Name: sort.out
Source File Name: sort.xxx

Meta-Loopless Sorts

Background

Sorting holds an important place in computer science. Analyzing and implementing various sorting algorithms forms an important part of the education of most computer scientists, and sorting accounts for a significant percentage of the world's computational resources. Sorting algorithms range from the bewilderingly popular Bubble sort, to Quicksort to parallel sorting algorithms and sorting networks. In this problem you will be writing a program that creates a sorting program (a meta-sorter).

Problem Description

The problem is to create a program whose output is a standard Pascal program that sorts *n* numbers where *n* is the only input to the program you will write. The Pascal program generated by your program must have the following properties:

- It must begin with `program sort (input, output) ;`
- It must declare storage for exactly *n* `integer` variables. The names of the variables must come from the first *n* letters of the alphabet (a,b,c,d,e,f).
- A single `readln` statement must read in values for all the integer variables.
- Other than `writeln` statements, the only statements in the program are `if then else` statements. The boolean conditonal for each `if` statement must consist of one strict inequality (either < or >) of two integer variables. Exactly *n*! `writeln` statements must appear in the program.
- Exactly three semi-colons must appear in the program
    1. After the program header: `program sort (input, output) ;`
    2. After the variable declaration: `... : integer ;`
    3. After the readln statement: `readln (... ) ;`
- No redundant comparisons of integer variables shoud be made. For example, during program execution, once it is determined that $a < b$, variables *a* and *b* shoud not be compared again.
- Every `writeln` statement must appear on a line by itself.
- The program must compile. Executing the program with input consisting of any arrangement of any n distinct integer values should result in the input values being printed in sorted order.

For those unfamiliar whith Pascal syntax, the example at the end of this problem completely defines the small subset os Pascal needed.

Input

The input is the single integer n on a line by itself whith $1 \le n \le 6$.

Output

The outpu is a compatible standard Pascal program meeting the criteria specified above.

Sample Input

3

Sample Output

```
program sort (input,output );
var
a,b,c : integer;
begin
     readln (a,b,c);
     if a < b then
          if b < c then
               writeln (a,b,c)
          else if a < c then
               writeln (a,c,b)
          else
               writeln (c,a,b)
     else
          if a < c then
               wirteln (b,a,c)
          else if b < c then
               writeln (b,c,a)
          else
               writeln (c,b,a)
end.
```

lnput File: quadtrees.in
Output File: quadtrees.out
Source File: quadtrees.xxx

Quadtrees

Problem Description

A quadtree is a representation format used to encode images. The fundamental idea behind the quadtree is that any image can be split into four quadrants. Each quadrant may again be split in four sub quadrants, etc. In the quadtree, the image is represented by a parent node, while the four quadrants are represented by four child nodes, in a predetermined order.

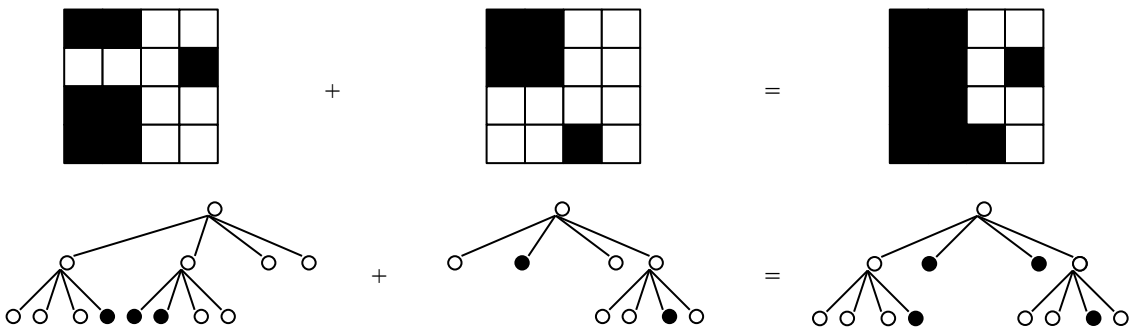
Of course, if the whole image is a single color, it can be represented by a quadtree consisting of a single node. In general, a quadrant needs only to be subdivided if it consists of pixels of different colors. As a result, the quadtree need not be of uniform depth.

A modern computer artist works with black-and-white images of 32 x 32 units, for a total of 1024 pixels per image. One of the operations he performs is adding two images together, to form a new image. In the resulting image a pixel is black if it was black in at least one of the componet images, otherwise it is white.

This particular artist believes in what he calls the *preferred fullness:* for an image to be interesting (i.e. to sell for big bucks) the rnost important property is the number of filled (black) pixels in the image So, before adding two images together, he would like to know how many pixels will be black in the resulting image. Your job is to write a program that, given the quadtree representation of two images, calculates the number of pixels that are black in the image, which is the restult of adding the two images together.

In the figure, the first example is shown (from top to bottom) as image, quadtree, preorder string (defined below) and number of pixels. The quadrant numbering is shown at the top of the figure.

2	1
3	4



ppeeefpffeefe	+	pefepeefe	=	ppeeefffpeefe
448	+	320	=	640

Input

The first line of input specifies the number of test cases (*N*) your program has to process. The input for each test case is two strings, each string on its own line. The string is the pre-order representation of a quadtree, in which the letter 'p' indicates a parent node, the letter 'f'(full) a black quadrant and the letter 'e'(emty) a white quadrant. It is guaranteed that each string represents a valid quadtree, while the depth of the tree is not more than 5 (because each pixel has only one color).

Output

For each test case, print on one line the text 'There are *X* black pixels.', where *X* is the number of black pixels in the resulting image.

Sample Input

```
3
ppeeefpffeefe
pefepeefe
peeef
peefe
peeef
peepefefe
```

Sample Input

```
There are 640 black pixels.
There are 512 black pixels.
There are 384 black pixels.
```

University of Puerto Rico
Mayaguez Campus

ICOM Challenge 2000

Intermediate Division

Sponsored by

***AEIC* and Lucent Technologies**

Table of Contents

Problem **Page**

ICOM Challenge Programming Contest
March 4, 2000
Intermediate Division

Input File: packets.in
Output File: packets.out
Source File: packets.xxx

Packets

Problem Description

A factory produces products packed in square packets of the same height *h* and of the sizes 1x1, 2x2, 3x3, 4x4, 5x5, 6x6. These products are always delivered to customers in the square parcels of the same height *h* as the products have and of the size 6x6. Because of the expenses it is the interest of the factory as well as of the customer to minimize the number of parcels necessary to deliver the ordered products from the factory to the customer. A good program solving the problem of finding the minimal number of parcels necessary to deliver the given products according to an order would save a lot of money. You are asked to make such a program.

Input

The input file consists of several lines specifying orders. Each line specifies one order. Orders are described by six integers separated by one space representing successively the number of packets of individual size from the smallest size 1x1 to the biggest size 6x6. The end of the input file is indicated by the line containing six zeros.

Output

The output file contains one line for each line in the input file. This line contains the minimal number of parcels into which the order from the corresponding line of the input file can be packed. There is no line in the output file corresponding to the last "null" line of the input file.

Sample Input

```
0    0    4    0    0    1
7    5    1    0    0    0
0    0    0    0    0    0
```

Sample Output

```
2
1
```

ICOM Challenge Programming Contest
March 4, 2000
Intermediate Division

Input File: tangle.in
Output File: tangle.out
Source File: tangle.xxx

Telephone Tangles

Problem Description

A large company wishes to monitor the cost of phone calls made by its personnel. To achieve this the PABX (Private Automatic Branch Exchange) logs, for each call, the number called (a string of up to 15 digits) and the duration in minutes. Write a program to process this data and produce a report specifying each call and its cost, based on standard Telecom charges.

International (IDD) numbers start with two zeroes (00) followed by a country code (1-3 digits) followed by a subscriber's number (4-10 digits). National (STD) calls start with one zero (0) followed by an area code (1-5 digits) followed by the subscriber's number (4-7 digits). The price of a call is determined by its destination and its duration. Local calls start with any digit other than 0 and are free.

Input

Input will be in two parts. The first part will be a table of IDD and STD codes, localities and prices as follows:

Code △ Locality nameSprice in cents per minute

where △ represents a space. Locality names are 25 characters or less. This section is terminated by a line containing 6 zeroes (000000).

The second part contains the log and will consist of a series of lines, one for each call, containing the number dialled and the duration. The file will be terminated a line containing a single #. The numbers will not necessarily be tabulated, although there will be at least one space between them. Telephone numbers will not be ambiguous.

Output

Output will consist of the called number, the country or area called, the subscriber's number, the duration, the cost per minute and the total cost of the call, as shown below.

Local calls are costed at zero. If the number has an invalid code, list the area as "Unknown" and the cost as -1.00.

Sample Input

```
088925 Broadwood$81
03 Arrowtown$38
0061 Australia$140
000000
031526         22
0061853279     3
0889256287213  122
779760 1
002832769      5
#
```

Sample Output

```
 031526         Arrowtown          1526        22        0.38    8.36
 0061853279     Australia        853279         3        1.40    4.20
 0889256287213  Broadwood       6287213       122        0.81   98.82
 779760         Local            779670         1        0.00    0.00
 002832769      Unknown                        5               -1.00
```

ICOM Challenge Programming Contest March 4, 2000 Intermediate Division

Input File: huffman.in
Output File: huffman.out
Source File: huffman.xxx

Variable Radix Huffman Encoding

Problem Description

Huffman encoding is a method of developing an optimal encoding of the symbols in a *source alphabet* using symbols from a *target alphabet* when the frequencies of each of the symbols in the source alphabet are known. Optimal means the average length of an encoded message will be minimized. In this problem you are to determine an encoding of the first N uppercase letters (the source alphabet, S_1 through S_N, with frequencies f_1 through f_n) into the first R decimal digits (the target alphabet, T_1 through T_R).

Consider determining the encoding when R=2. Encoding proceeds in several passes. In each pass the two source symbols with the lowest frequencies, say S_1 and S_2, are grouped to form a new "combination letter" whose frequency is the sum of f_1 and f_2. If there is a tie for the lowest or second lowest frequency, the letter occurring earlier in the alphabet is selected. After some number of passes only two letters remain to be combined. The letters combined in each pass are assigned one of the symbols from the target alphabet.

The letter with the lower frequency is assigned the code 0, and the other letter is assigned the code 1. (If each letter in a combined group has the same frequency, then 0 is assigned to the one earliest in the alphabet. For the purpose of comparisons, the value of a "combination letter" is the value of the earliest letter in the combination.) The final code sequence for a source symbol is formed by concatenating the target alphabet symbols assigned as each combination letter using the source symbol is formed.

The target symbols are concatenated in the reverse order that they are assigned so that the first symbol in the final code sequence is the last target symbol assigned to a combination letter.

The two illustrations below demonstrate the process for *R*=2.

Symbol	Frequency
A	5
B	7
C	8
D	15

Pass 1: A and B grouped
Pass 2: (A,B) and C grouped
Pass 3: (A,B,C) and D grouped
Resulting codes: A=110, B=111, C=10, D=10
Avg. Length=(3*5+3*7+2*8+ 1*15)/35=1.91

Symbol	Frequency
A	7
B	7
C	7
D	7

Pass 1: A and B grouped
Pass 2: C and D grouped
Pass 3: {A,B} and {C,D} grouped
Resulting codes: A=00, B=01, C=10, D=11
Avg. Length=(2*7+2*7+2*7+2*7)/28=2.00

When *R* is larger than 2, *R* symbols are grouped in each pass. Since each pass effectively replaces *R* letters or combination letters by 1 combination letter, and the last pass must combine *R* letters or combination letters, the source alphabet must contain *k*(R-1)+R* letters, for some integer k.

Since *N* may not be this large, an appropriate number of fictitious letters with zero frequencies must be included. These fictitious letters are not to be included in the output. In making comparisons, the fictitious letters are later than any of the letters in the alphabet.

Now the basic process of determining the Huffman encoding is the same as for the *R=2* case. In each pass, the *R* letters with the lowest frequencies are grouped, forming a new combination letter with a frequency equal to the sum of the letters included in the group. The letters that were grouped are assigned the target alphabet symbols 0 through *R*-1.0 is assigned to the letter in the combination with the lowest frequency, 1 to the next lowest frequency, and so forth. If several of the letters in the group have the same frequency, the one earliest in the alphabet is assigned the smaller target symbol, and so forth.

The illustration below demonstrates the process for R=3.

Symbol	Frequency
A	5
B	7
C	8
D	15

Pass 1: ? (fictitious symbol), A and B are grouped
Pass 2: (?,A,B}, C and D are grouped
Resulting codes: A=1 1, B=12, C=0, D=2
Avg. Length=(2*5+2*7+ 1 *8+1 *15)/35=1.34

Input

The input will contain one or more data sets, one per line. Each data set consists of an integer value for *R* (between 2 and 10), an integer value for *N* (between 2 and 26), and the integer frequencies f_1 through f_N, each of which is between 1 and 999.

The end of data for the entire input is the number 0 for *R*; it is not considered to be a separate data set.

Output

For each data set, display its number (numbering is sequential starting with 1) and the average target symbol length (rounded to two decimal places) on one line. Then display the *N* letters of the source alphabet and the corresponding Huffman codes, one letter and code per line. The examples below illustrate the required output format.

Sample Input

```
2 5 5 10 20 25 40
2 5 4 2 2 1 1
3 7 20 5 8 5 12 6 9
4 6 10 23 18 25 9 12
0
```

Sample Output

```
Set 1; average length    2.10
      A: 1100
      B: 1101
      C: 111
      D: 10
      E: 0



Set 2; average length    2.20
      A: 11
      B: 00
      C: 01
      D: 100
      E: 101



Set 3; average length 1.69
    A: 1
    B: 00
    C: 20
    D: 01
    E: 22
    F: 02
    G: 21
```

Set 4; average length 1.32

A: 32
B: 1
C: 0
D: 2
E: 31
F: 33

University of Puerto Rico
Mayaguez Campus

ICOM Challenge 2000

Beginner Division

Sponsored by

AEIC and Lucent Technologies

Table of Contents

Problem **Page**

ICOM Challenge Programming Contest
March 4, 2000
Beginner Division

Input File: master, in
Output File: master.out
Source File: master.xxx

Master-Mind Hints

Problem Description

MasterMind is a game for two players. One of them, *Designer*, selects a secret code. The other, *Breaker*, tries to break it. A code is no more than a row of colored dots. At the beginning of a game, the players agree upon the length *N* that a code must have and upon the colors that may occur in a code.

In order to break the code, Breaker makes a number of guesses, each guess itself being a code. After each guess Designer gives a hint, stating to what extent the guess matches his secret code.

In this problem you will be given a secret code $s_1 \ldots s_n$ and a guess $g_1 \ldots g_n$, and are to determine the hint. A hint consists of a pair of numbers determined as follows.

A match is a pair (i,j), $1 \leq i \leq n$ and $1 \leq j \leq$ n, such that $s_i = g_j$. Match (i,j) is called *strong* when $i = j$, and is called *weak* otherwise. Two matches (i,j) and (p,q) are called independent when $i = p$ if and only if $j = q$. A set of matches is called *independent* when all of its members are pairwise independent. The following table is an example of how a match pair is given by the *Designer* of the secret code to the *Breaker*.

Table 1: Designer Code 1355.

Breaker Guess Number	Breaker Guess Code	Match Pair (*i*,*j*)
1	1123	(1,1)
2	4335	(2,0)
3	6551	(1,2)
4	6135	(1,2)
5	1355	(4,0)

Designer chooses an independent set M of matches for which the total number of matches and the number of strong matches are both maximal. The hint then consists of the number of strong followed by the number of weak matches in *M*. Note that these numbers are uniquely determined by the secret code and the guess. If the hint turns out to be (*n*,0), then the guess is identical to the secret code.

Input

The input will consist of data for a number of games. The input for each game begins with an integer specifying *N* (the length of the code). Following these will be the secret code, represented as *N* integers, which we will limit to the range 1 to 9. There will then follow an arbitrary number of guesses, each also represented as *N* integers, each in the range 1 to 9. Following the last guess in each game will be *N* zeroes, these zeroes are not to be considered as a guess. Following the data for the first game will appear data for the second game (if any) beginning with a new value for *N*. The last game in the input will be followed by a single zero (when a value for *N* would normally be specified). The maximum value for *N* will be 1000.

Output

The output for each game should list the hints that would be generated for each guess, in order, one hint per line. Each hint should be represented as a pair of integers enclosed in parentheses and separated by a comma. The entire list of hints for each game should be prefixed by a heading indicating the game number; games are numbered sequentially starting with 1. Look at the samples below for the exact format.

Sample Input

```
4
1 3 5 5
1 1 2 3
4 3 3 5
6 5 5 1
6 1 3 5
1 3 5 5
0 0 0 0
10
1 2 2 2 4 5 6 6 6 9
1 2 3 4 5 6 7 8 9 1
1 1 2 2 3 3 4 4 5 5
1 2 1 3 1 5 1 6 1 9
1 2 2 5 5 5 6 6 6 7
0 0 0 0 0 0 0 0 0 0
0
```

Sample Output

```
Game      1:
     (1,1)
     (2,0)
     (1,2)
     (1,2)
     (4,0)
Game 2:
    (2,4)
    (3,2)
    (5,0)
    (7,0)
```

March 4, 2000
Beginner Division

Input File: isbn.in
Output File: isbn.out
Source File: isbn.xxx

Recognizing Good ISBNs

Problem Description

Most books now published are assigned a code which uniquely identifies the book. The International Standard Book Number, or ISBN, is normally a sequence of 10 decimal digits, but in some cases, the capital letter X may also appear as the tenth digit. Hyphens are included at various places in the ISBN to make them easier to read, but have no other significance. The sample input and expected output shown below illustrate many valid, and a few invalid, forms for ISBNs.

Actually, only the first nine digits in an ISBN are used to identify a book. The tenth character serves as a check digit to verify that the preceding 9 digits are correctly formed. This check digit is selected so that the value computed as shown in the following algorithm is evenly divisible by 11. Since the check digit may sometimes need to be as large as 10 to guarantee divisibility by 11, a special symbol was selected by the ISBN designers to represent 10, and that is the role played by X.

The algorithm used to check an ISBN is relatively simple. Two sums, *s*l and *s*2, are computed over the digits of the ISBN, with *s*2 being the sum of the partial sums in *s*l after each digit of the ISBN is added to it. The ISBN is correct if the final value of *s*2 is evenly divisible by 11.

An example will clarify the procedure: Consider the (correct) ISBN 0-13-162959-X (for Tanenbaum's Computer Networks). First look at the calculation of sl:

Digits in the ISBN	0	1	3	1	6	2	9	5	9	10 (X)
sl (partial sums)	0	1	4	5	11	13	22	27	36	46

The calculation of *s*2 is done by computing the total of the partial sums in the calculation of *s*1:

*s*2 (running totals)	0	1	5	10	21	34	56	83	119	165		

We now verify the correctness of the ISBN by noting that 165 is, indeed, evenly divisible by 11.

Input

The input data for this problem will contain one candidate ISBN per line of input, perhaps preceded and/or followed by additional spaces. No line will contain more than 80 characters, but the candidate ISBN may contain illegal characters, and more or fewer than the required 10 digits. Valid ISBNs may include hyphens at arbitrary locations. The end of file marks the end of the input data.

Output

The output should include a display of the candidate ISBN and a statement of whether it is legal or illegal. The expected output shown below illustrates the expected form.

Sample Input

```
0-89237-010-6
0-8306-3637-4
   0-06-017758-6
   This_is_garbage
1-56884-030-6
     0-8230-2571-3
     0-345-31386-0
     0-671-88858-7
     0-8104-5687-7
     0-671-74119-5
     0-812-52030-0
     0-345-24865-1-150
0-452-26740-4
     0-13-139072-4
     0-1315-2447-X
```

Sample Output

```
0-89237-010-6 is correct.
0-8306-3637-4 is correct.
0-06-017758-6 is correct.
This_is_garbage is incorrect.
1-56884-030-6 is correct.
0-8306-3637-4 is correct.
0-06-017758-6 is correct.
1-56884-030-6 is correct.
0-8230-2571-3 is correct.
0-345-31386-0 is correct.
0-671-88858-7 is correct.
0-8104-5687-7 is correct.
0-671-74119-5 is correct.
0-812-52030-0 is correct.
0-345-24865-1-150 is incorrect.
0-452-26740-4 is correct.
0-13-139072-4 is correct.
0-1315-2447-X is correct.
```

ICOM Challenge Programming Contest
March4, 2000
Beginner Division

Input File: packets.in
Output File: packets.out
Source File: packets.xxx

Packets

Problem Description

A factory produces products packed in square packets of the same height *h* and of the sizes 1x1, 2x2, 3x3, 4×4, 5x5, 6x6. These products are always delivered to customers in the square parcels of the same height *h* as the products have and of the size 6x6. Because of the expenses it is the interest of the factory as well as of the customer to minimize the number of parcels necessary to deliver the ordered products from the factory to the customer. A good program solving the problem of finding the minimal number of parcels necessary to deliver the given products according to an order would save a lot of money. You are asked to make such a program.

Input

The input file consists of several lines specifying orders. Each line specifies one order. Orders are described by six integers separated by one space representing successively the number of packets of individual size from the smallest size 1x1 to the biggest size 6x6. The end of the input file is indicated by the line containing six zeros.

Output

The output file contains one line for each line in the input file. This line contains the minimal number of parcels into which the order from the corresponding line of the input file can be packed. There is no line in the output file corresponding to the last "null" line of the input file.

Sample Input

```
0 0 4 0 0 1
7 5 1 0 0 0
0 0 0 0 0 0
```

Sample Output

```
2
1
```

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '99
Expert Division

Sponsored by

***AAAEIC* and Lucent Technologies**

Table of Contents

Problem **Page**

ICOM Challenge Programming Contest

March 13, 1999

Expert Division

Input File: decimals.in

Output File: decimals.out

Source File: decimals.<xxx>

Awesome Decimals

Problem Description

Dare to find a string that "adds up" to a given number. The string will consist of the decimal digits (0-9) with optional plus (+) and minus (-) signs between them. Digits between two signs form a single decimal number. What's the catch? You have to use all decimal digits, in ascending order. Read the sample input and you will see.

Your program will read several integers from the input file, one number per line. The output file will have one line per each line of the input file. For each number with a valid solution, the line will have the solution string followed by a space, an equal (=) sign, another space and the number given. If there is no solution for that number, the line will have the string "There is no solution for" followed by the number. Remember to print the "+" sign at the beginning when required.

Sample Input

```
23456788
9843574
45
876
```

Sample Output

```
-01+23456789 = 23456788
There is no solution for 9843574
+01-23+45-67+89 = 45
-01-2+34+56+789 = 876
```

ICOM Challenge Programming Contest

March 13, 1999

Expert Division

Input File: cache.in

Output File: cache.out

Source File: cache.<xxx>

To Cache or not to Cache . . .

Problem Description

Your job is to make a simulation of a memory system, and to gather statistics on its performance. You will use actual memory traces from the software that will run on the system. The system has a relatively small but fast cache memory and a large but slow RAM. The address space is 32 bits wide, all of which is byte-addressable. The cache is a direct mapped cache.

The input file will have the following information:

- Number of bytes in normal memory
- Number of bytes in cache memory
- Number of bytes per cache line
- Memory accesses of a portion of the code

You can assume the following about the input data:

- All measures of the cache and memory (first four inputs) are powers of two
- I < cache size < Memory size < 4GB
- The cache can hold at least one complete set of lines.
- Only one processor (bus master) has access to this memory system.

The output will have a line per memory access included in the input file. Your program shall output if the access resulted on a miss or a hit, and what line of the cache was accessed.

Hints:

- See the last page of this problem for a refresher course on caches

Sample Input

```
8
4
2
3  2  1 3  0  4  4  4  5  5  6  3  3  3  5  6  7  4  3  7  6  5  1  1  4  5  4  6  7
```

Sample Output

```
Miss 0
Miss 1
Hit 1
Hit 0
Hit 1
Hit 0
Miss 0
Hit 0
Hit 0
Hit 0
Hit 0
Miss 1
Miss 1
Hit 1
Hit 1
Hit 0
Miss 1
Hit 1
Hit 0
Miss 1
Miss 1
Hit 1
Hit 0
Miss 0
Hit 0
Miss 0
Hit 0
Hit 0
Hit 1
Hit 1
30
10
20
0.66666
```

Cache lines

Caches are divided in sections called *lines*. A cache line can hold one or more RAM locations, but on a given cache this number is constant. Those locations are always contiguous. When data from RAM are brought to the cache, enough data to fill a whole line is copied, even though only one location may be wanted at the time. These other locations may be referenced in subsequent accesses without penalty until the line is replaced by another line from RAM.

Knowing what's in a cache line

A tag is kept or each line saved in the cache. Each tag will serve to determine if the RAM location of the current access is part of its corresponding line. Tags are constructed by saving the most significant bits of the addresses of the locations of a line. All locations in a line will share the same address prefix. The least significant bits of the address determine the offset of each location from the start of the line.

Associativity

The cache may be divided further into groups of lines called sets. Every set has the same size. Each set corresponds to a group of lines of RAM. Each line on this group can be placed in any of the lines of the corresponding set.

- Direct mapped cache - When each set has only one line. There is only one possible cache line for each location of memory.
- Fully associative cache - The whole cache is formed by a single set. Data from RAM may be found in any line of the cache.
- Set associative cache - There is more than one set in the cache, and each set has n lines. A line from RAM may be placed on any of the n lines of the corresponding set. This cache is said to be n-way set associative.

Note that he first two types of caches are special cases of set associative caches.

Computing set number and tags

The figure below shows haw addresses are decomposed to give their corresponding tag, set and block offset. Note that:

- Number of bits needed for block offset = Iog2(cache line size)
- Number of bits needed for set number = Iog2(number of sets)
- The rest of the address may be used for the tag

m-bit address

Tag	Set	Block Offset

ICOM Challenge Programming Contest
March 13, 1999
Expert Division

Input File: obstacle.in
Output File: obstacle.out
Source File: obstacle.<xxx>

Obstacles

Problem Description

A robot moves in a room full of obstacles. Given the present position of the robot, find the shortest path to a given destination, avoiding any obstacles. You may assume the following:

- The robot and the destination have no size.
- All obstacles are rectangular.
- There is one and only one shortest valid route to the destination.
- The room is a square of maximum size 40
- Obstacles do not touch one another
- All coordinates are given in integer numbers
- There will be no more than ten (10) obstacles in any input set.

The input file contains sequence of independent input sets. Each set has the following information:

- The initial position of the robot (x-y coordinates)
- The destination point (x-y coordinates)
- The number of obstacles
- For each obstacle (one line per obstacle):
  - The x-coordinate of the left side
  - The x-coordinate of the right side
  - The y-coordinate of the top side
  - The y-coordinate of the bottom side

The output consists of a sequence of x-y coordinates that represent successive positions of the robot between straight-line moves, followed by the distance traveled by the robot over the shortest path. If there is more than one path with the same shortest distance, chose the one with the lowest number of moves. Your program should find the answer in a reasonable amount of time. Ask the judges for instructions on how much time is reasonable.

Sample Input

```
1  1
9  9
10
1  2  4  3
3  4  8  3
5  10  3  1
9  10  5  4
5  10  7  6
6  8  9  8
10  11  9  8
10  11  11  10
9  8  11  10
3  4  10  9
```

Sample Output

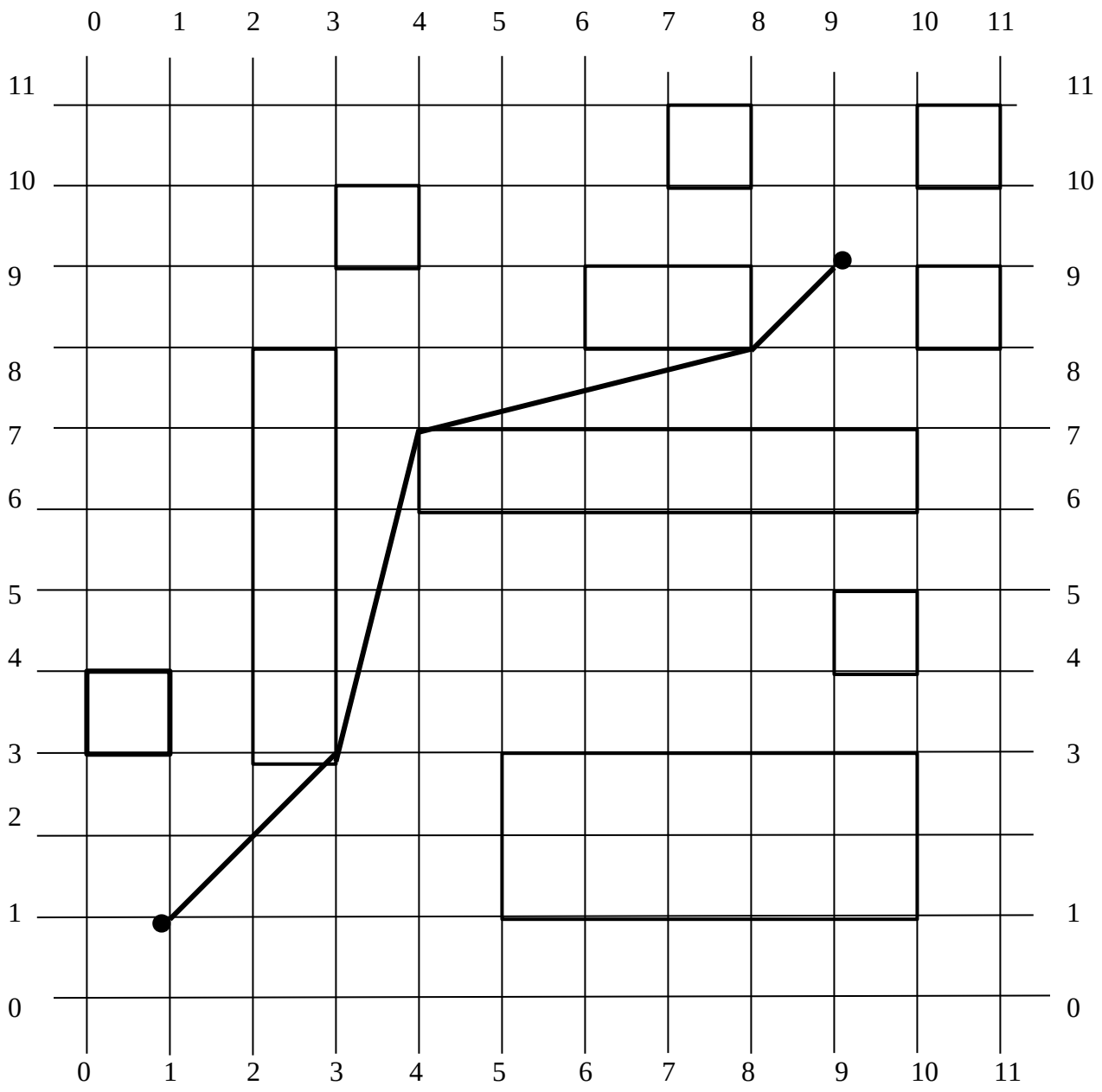
```
(1, 1),  (4, 3),  (5, 7),  (8, 8),  (9, 9)
D: 12.3051
```

Hints:

- Consider representing each obstacle by a system of four inequalities. Each movement may be represented as a segment. If any point of the segment satisfies the inequalities, then the move is not valid, because it crosses an obstacle.
- An alternative would be to represent the problem space using a directed graph, and then use a well-known algorithm for finding the single-source shortest path on the directed graph.
- Note that this problem becomes untractable (by using only brute force) for more than a few number of obstacles (O((n +4)!), for *n* obstacles). After having your program work for a few obstacles, implement a simple heuristic to prune the search tree; i. e. during the search, eliminate paths that are obviously not better than the best path so far.

Notes:

Here is a picture for the sample input and output shown above.



ICOM Challenge Programming Contest

March 13, 1999

Expert Division

Input File: interprt.in
Output File: interprt.out
Source File: interprt.<xxx>

A Simple Interpreter

Problem Description

The source code for programs written on a certain programming language tends to be rather large. This is because the language is based on operations for a very simple virtual stack machine. All data is saved on the stack and all operations use the stack for parameters as well as for results. A program consists of a sequence of one or more of the following operations:

`push<literal>`	- pushes a literal (integer) onto the stack
`pop`	- pops
`add`	- adds the top two values on the top of the stack
`sub`	- subtracts the value on top of the stack from the value below it
`mult`	- multiplies the top two values on the top of the stack
`print`	- displays the value on top of the stack

This problem may be alleviated by adding more powerful constructs to the language. For instance, consider adding the following:

`for`	- Marks the start of a loop. Takes two arguments from the stack: Initial (I) and final (f) values of the index of the loop.
`next`	- Marks the end of the current loop
`push <offset>`	- Pushes an index offset into the stack

Loops may be nested. Indexes of outer loops may be accessed from nested loops by specifying the number of loops between them. The syntax for <offset> is $<number>, where <number> is the number of loops between the current loop and the loop of the index referenced. For example, in the following sequence (line numbers are not part of the code):

```
01        push 1
02        push 10
03        for
04        push 1
5     push $0
6     for
7     push $0
8     push $1
9     mult
10    print
11    next
12    next
```

line 05 will reference the index of the loop that starts in line 03, while line 07 will reference the index of the loop that starts in line 06. Line 08 will reference the index of the loop that starts in line 03.

Write a program that will interpret programs written in the new language. Your interpreter should detect the following errors:

- An index offset out of bounds (there exists no loop index at the nesting level specified) Print: "Offset out of bounds" on a single line.
- Invalid literal (all literals should be unsigned integers < 2_{16}-1) Print: "Invalid literal specified" on a single line.
- Invalid syntax. Print: "Syntax error" on a single line.

If an error is detected, your interpreter should assume that the rest of the program is a valid program on its own.

Sample Input

```
push 1
push 100
mult 34
push
push 66539
push 1
push 10
for
   push $1
push 1
push 5
for
   push 1
   push $0
   for
      push 1
      push $0
      mult
      print
   next
next
```

Sintax Error: restart with next line

Invalid Literal: restart with next line

Offset out of bounds: restart with next line

This segment is a complete program

Sample Output

```
Syntax error
Invalid literal specified
Offset out of bounds
1
2
4
3
6
9
4
8
12
16
5
10
15
```

20
25

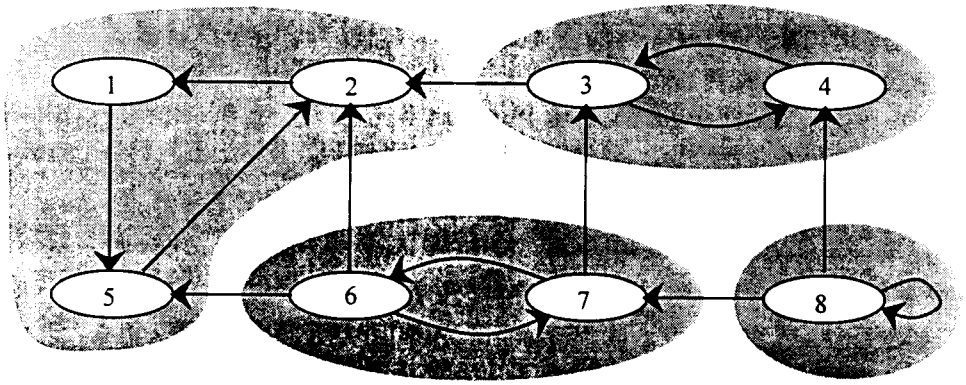
ICOM Challenge Programming Contest
March 13, 1999
Expert Division

Input File: graph.in
Output File: graph.out
Source File: graph.<xxx>

Strongly Connected Components

Problem Description

Code a program that determines the strongly connected components of a given directed graph. Recall that a strongly connected component of a directed graph$G = (V,E)$ is a maximal set of vertices $U \subseteq V$ such that for every pair of vertices u and v in U, we have both $u.\sim v$ and $v\ u$; that is, vertices u and v reachable from each other. The following is an example taken from *Introduction to Algorithms,* T.H. Cormen, C. E. Leiserson and R. L. Rivest.



Gray regions mark the strongly connected components of the graph.

The input contains a sequence of input sets, each containing the number of arcs that follow for that set, and the set E of directed arcs. There are no non-connected vertices. Therefore, the set of vertices V is implied by the arcs themselves.

For each input set, your program should output a list of vertices (one line per strongly connected component), each separated by a space. Leave one empty line between lists of components.

Sample Input

```
14
15
52
21
65
62
32
67
76
73
34
87
43
84
88
```

Sample Output

```
125
34
67
8
```

Note:

This input sample has only one set. Your program must accept multiple input sets from the same file.

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '99

Intermediate Division

Sponsored by

***AEIC* and Lucent Technologies**

Table of Contents

Problem **Page**

ICOM Challenge Programming Contest
March 13, 1999
Intermediate Division

Input File: dblog.in
Output File: dblog.out
Source File: dblog.<xxx>

Database Logging Tables

Problem Description

Most databases have a logging system that keeps track of all data modifications. The log generated by this system aids in the recovery of the database after a crash.

The data modifications are made through concurrent transactions that affect the database. A transaction reads data, UPDATEs the data and saves it in a space in temporary memory, called a page. The transaction requests that all data updates it has made are COMMITted (i.e. written to disk). When all that data is written to disk, the transaction has ENDed. In case of a database, system or disk error it is possible to ABORT a transaction. The logging system keeps a chronological list of these actions. Each list record is of the form:

LSN	Type	TransID	PageID

Each record has a unique log sequence number *(LSN).* The type is the action being executed by the transaction (update, commit, end or abort). The *transID* is the identification number for the transaction. The *pageID* is the identification number for the page being modified.

To provide a static image of the database state before a crash, the logging system also keeps track of active transactions and modified pages. Two tables are used for this purpose:

- A dirty page table (DPT) keeps track of the pages in use. It stores the *pageID* of the modified page and the *LSN,* called *recLSN,* of the first log record that caused the page to become dirty. Once the transaction that causes the page to become dirty ends or is aborted, the page record in the table is erased.

- A transaction table (TT) contains one entry for each active transaction. The entry contains the *transID,* the *LSN* of the most recent log record for this transaction (called *lastLSN)* and the *status* of the transaction that is in process (active, aborted or commited). Once the transaction is ended or aborted its record in the table is erased.

Given the log records, reconstruct the dirty page and transaction tables. Show the process that reconstructs the tables. Input file dblog.in contains the log records in the format:

```
LSN type: transID pageID
```

The output file dblog.out should contain the dirty page table and transaction table in the format:

```
LSN: DPT=[pageID,recLSN),...],TT=[(transID,lastLSN,status),..]
```

Notes

- A transaction is active if it has made an update and is not committed, aborted or ended.
- Only update records require *pageID*s.
- A page is not taken off the dirty page table until all transactions that modified it are either committed/ended or aborted.

Sample Input

```
10 update: T1 P1
20 commit: TI
30 end: TI
40 update: T2 P2
50 update: T3 P3
60 update: T2 P3
70 abort: T2
80 update: T4 P5
90 update: T4 P4
100 commit: T3
```

Sample Output

```
10: DPT=[(P1,10)], TT=[(T1, 10, a)]
20: DPT=[(P1,10)], TT=[(T1, 20, c)]
30: DPT=[], TT=[]
40: DPT=[(P2,40)], TT=[(T2, 40, a)]
50: DPT=[(P2,40), (P3,50)], TT=[(T2, 40, a), (T3,50,a)]
60: DPT=[(P2,40), (P3,50)], TT=[(T2, 60, a), (T3,50,a)]
70: DPT=[(P2,40), (P3,50)], TT=[(T2, 60, b), (T3,50,a)]
80: DPT=[(P3,50), (P5,80)], TT=[(T3,50,a), (T4,80,a)]
90: DPT=[(P3,50), (P5,80), (P4, 90)], TT=[(T3,50,a), (T4,90,a)]
100: DPT=[(P3,50), (P5,80), (P4, 90)], TT=[(T3,100,c), T4,90,a)]
```

ICOM Challenge Programming Contest
March 13, 1999
Beginner Division

Input File: subset.in
Output File: subset, out
Source File: subset.<xxx>

Subsets

Problem Description

Given a set of characters, find all its subsets. The input file contains a list of characters separated by spaces. The output will contain all the possible subsets; one subset per line.

Sample Input

```
a  b  c
```

Sample Output

```
{a b c d}
{a b c}
{a b d}
{a c d}
{b c d}
{a b}
{a c}
{a d}
{b c}
{b d}
{c d}
{a}
{b}
{c}
{d}
{}
```

ICOM Challenge Programming Contest
March 13, 1999
Intermediate Division

Word Puzzle ++

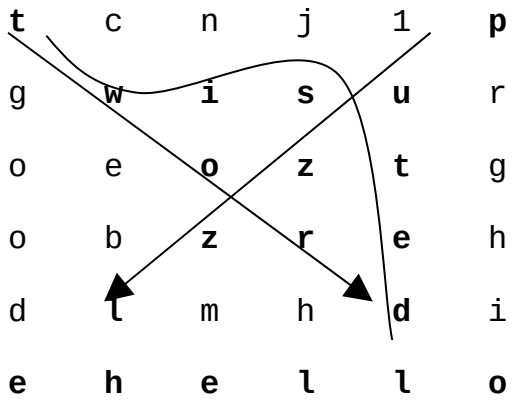
Input File: puzzle2.in
Output file: puzzle2.out
Source File: puzzle2.<xxx>

Puzzle Description

In a word puzzle you are given a list of words and a grid of letters. The objective of the game is to find all listed words embedded in the grid of letters. The words can be found horizontally, vertically or diagonally. In this variation of the game, words can also appear twisted (see figure below).

Make a program that will solve a word puzzle. Your program must find all words present and output the position of each letter of the word on the grid. The input file consists of the dimension of the grid, the grid of letters and a list of words to find.

The output file will be the list of letters, followed by the coordinate of every letter of the word, or the string "not found".



Sample Input

```
6
TCHNLP
GWISUR
```

```
OEOZTG
OBZREH
DLMHDI
EHELLO
puzzle
word
problem
hello
twisted
array
input
output
```

Sample Output

```
puzzle (0, 5) (1, 4) (2, 3) (3, 2) (4, 1)   (5, 0)
word (1, 1) (2, 2) (3, 3) (4, 4)
problem not found
hello (5, 1) (5, 2) (5, 3) (5, 4)
twisted (0, 0) (1, 1) (1, 2) (1, 3) (2, 4) (3, 4) (4, 4)
array not found
input not found
output not found
```

AEIC Challenge Programming Contest
March 13, 1999
Intermediate Division

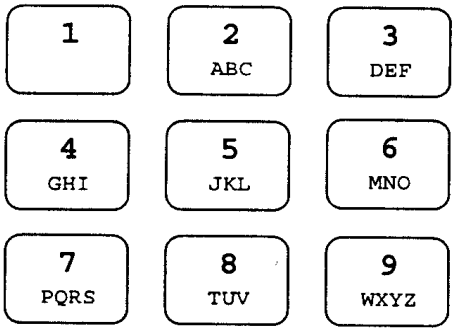
Input File: teldir. in
Output File: teldir. out
Source File: teldir.<xxx>

Telephone Directory Search

Problem Description

Some voice mail systems allow users to search for phone numbers of other registered users of the system. To do so, one must spell the name or a. prefix of the name of the person to call. This is accomplished by pressing the key number corresponding to each letter of the prefix.

Use the following diagram to map numbers on the keypad to letters.



Write a program that finds all names in the directory that match a given name prefix. The prefix is given as a sequence of keypad presses. Begin by matching the last name and then the first name.

The input will consist of:

- Number of names/records in the dictionary
- List of names in the dictionary
- Number of keypad numeric sequences
- Numeric sequences (one per line).

The maximum number of characters in the number sequence is 11. Allow 32 bytes for names.

The output file consists of the numeric sequence followed by the list of names that matched the sequence or the message "No Matches Found".

Sample Input

```
10
Velez, Carlos
Torres, Ana
Jolie, Angelina
Lopez, Maribel
Laguerre, Tony
Loperena, Martha
Santos, Benjamin
Korg, James
Klinger, Heidi
Questell, Eduardo
4
2345
567
56737
5
```

Sample Output

```
Sequence: 2345
Result:
No Matches Found

Sequence: 567
Results:
Lopez, Maribel
Loperena, Martha
Korg, James

Sequence: 56737
Results:
Loperena, Martha

Sequence: 5
Results:
Jolie, Angelina
Lopez, Maribel
Laguerre, Tony
Loperena, Martha
Korg, James
Klinger, Heidi
```

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '99

Beginner Division

Sponsored by

***AEIC* and Lucent Technologies**

Table of Contents

Problem **Page**

Input File: parity.in
Output File: parity.out
Source File: parity.<xxx>

Parity Checking

Problem Description

Perhaps the simplest form of transmission error detection is parity checking. The parity of a set of data is the parity of the number of bits with value of "1" in the data. For example, the data "00110010" has an odd parity, while "01110010" has an even parity.

In a transmission system, the sender of the data will bundle each data packet with an additional bit that will force the parity of the whole to be even or odd, depending on what parity the receiver and sender have agreed upon. The receiver will then assume that all correct packets have that parity. Any packets with incorrect parity will be discarded and resent.

The drawback of this method is that no more than one error may be detected. Furthermore, when the error is detected, the data cannot be corrected. The packet has to be resent from the other side of the transmission channel.

By using two-dimensional parity checking, more than one error may be detected. If there is only one error, the data may be corrected, avoiding resends. In two-dimensional parity checking, a group of data and parity bit packets are bundled together and an additional parity packet is generated using each of the packets in the group. The additional packet has parity bits for each column (bit position) of the data packets and their corresponding parity bits. The following is a group of data and parity bit packets followed by a parity packet.

Packet Number	Data (four bits)				Parity Bit
0	0	0	1	1	0
1	1	0	1	0	0
2	0	1	0	0	1
3	0	0	1	0	1
4	1	1	1	1	0

Your program will detect errors in a set of packets. If there is only one error, it will correct the error and show where the error occurred. You may assume the parity packet will never be corrupted.

The input will consist of a sequence of input sets. Each input set will correspond to a set of packets. For each input set, your program will read:

- The number of packets (n) on this set
- The length (in bits) of the packets
- N packets

For each input set, your program will show the following:

- The sequence of packets on the set (if there was only one error, the corresponding bit must be corrected and shown corrected)
- If more than one error is detected, mark the rows and columns were the error are with an asterisk to the right or bottom, respectively.

Sample Input

```
5
6
001100
111010
010010
001010
111100
3
4
0011
0101
0110
4
3
011
111
110
000
```

Sample Output

```
001100
111010
010010
001010
111100
    *     *
0011
0101
0110
```

```
011
101
110
000
Error corrected in packet 1, bit 1.
```

Input File: xmorse.in
Output File: xmorse.out
Source File: xmorse.<xxx>

XMorse

Problem Description

FBI Agents Fox Mulder and Dana Scully are following a new lead in their most recent X-Files case. They just received a top secret data file through unofficial channels that can provide them with very valuable information about the case they are working on. The information in the file appears to be encrypted with a series of '.' and '-'. The Agents believe that the message is encoded using a graphical representation of Morse code. Morse code represents characters of an alphabet as sequences of dits (short key closures) and dahs (longer key closures). If we let a period (.) represent a dit and a dash (-) represent a dah, then the Morse code version of the English alphabet can be represented as:

A	.-	N	-.
B	-...	O	---
C	-.-.	P	.--.
D	-..	Q	--.-
E	.	R	.-.
F	..-.	S	...
G	--.	T	-
H	….	U	..-
I	..	V	...-
J	.---	W	.--
K	-.-	X	-..-
L	.-..	Y	-.--
M	--	Z	--..

Help Agents Mulder and Scully to decode the secret message by writing a program that will interpret the information in Morse code as English.

The input file contains periods and dashes representing a message composed only of the English alphabet as specified above. One blank space is used to separate letters and three blanks are used to separate words.

Your program must output the English Interpretation of the Morse code found in the input file. There must be no blanks between letters of the same word in the output, and only one blank between words. You must use all capital letters in the output.

Sample Input

```
.... . .-.. .-.. --- -- -.-- -. .- -- . .. ... ... .- --
-.... .  - .-. ..-- .... .. ... --- .. - -  - .... . .-.
.
```

Sample Output

```
HELLO MY NAME IS SAM THE TRUTH IS OUT THERE
```

Input File: y2k.in
Output File: y2k.out
Source File: y2k.<xxx>

Y2K Problem

Problem Description

The Year 2000 problem stems from the fact that, in many computer systems and databases, the *year* component of the date is represented by two decimal digits. If a person's date of birth (in months/day/year format) is 04/25/92, the apparent age of the person would be 6 years old. But was the person born in 1892 or 1992?

Here are some rules that can be used to determine a person's age:

Rule 1: All persons in the input data will have been born on or before the date of this contest (March 13, 1999), this means:

- If the last two digits of the year of birth are 99 and the date within that year is in the range from March 14 through December 31 1999, then the year of birth is 1899.

Rule 2: If the apparent age is 20 or more, the first two digits of the year of birth are 19.

Rule 3: If the apparent age is in the range of 7-19, then the existence of school records will determine the first two digits of the year. If the person attended school during the past ten years, then the first two digits of the year of birth are 19, otherwise they are 18.

Rule 4: If the apparent age is 6 or less, then the existence of tax records will determine the first two digits of the year of birth. If there is a tax record for the person, the first two digits of the year of birth are 18; otherwise they are 19.

You may assume that all cases that arise in the input data will be covered by the rules just stated.

Each line of the input consists of seven fields. Adjacent fields are separated by one blank space. The meaning of each field of the seven fields of inputs follows:

Columns 1-25: the person's name

Columns 27-35: the person social security number

Columns 37-40: a 4 character code which designates the last known school attended by the person during the past ten years. The code "----" means that the person did not attend school during that period.

Columns 42-43: the last two digits of the most recent year in which the person filed a tax return, if a tax return is known to have been filed since 1950. Otherwise, there is no tax record for this person, column 42 will be blank and column 43 will contain the single digit 0.

Columns 45-46: the person's month of birth

Columns 48-49 the day of the month of the person's date of birth

Columns 51-52: the last two digits of the person's year of birth (or just the last two digits, if the person was born in the period 1900-1909)

If the data in columns 45-46, or columns 48-49, or columns 51-52 requires only one digit, it will be right-justified in the stated columns. Your program will write its output to the file y2k.out Each line in the input will give rise to an almost identical line of output, the only differences been that the two-digit year of birth from columns 51-52 will be replaced by a four-digit year of birth in columns 51-54. The first two digits of the year of birth will be 18 or 19, determined by the set of rules provided.

Sample Input

```
Dijkstra, Edgar            603849562 ----    0  9  1 99
Hopper, Grace Murray       234258030 ----   97  2 12  0
Hollerith, Herman          60234874  FGSH   97  5 20 77
Shannon, Claude            571839480 FGJH   96 10 28 79
Lovelace, Ada              183099831 BAMS    0  6 30 92
Babbage, Charles           419309823 ----    0  8 31 92
```

Sample Output

```
Dijkstra, Edgar            603849562         0  9 1  1899
Hopper, Grace Murray       234258030        97  2 12 1900
Hollerith, Herman          602348748 FGSH   97  5 20 1977

Shannon, Claude            571839480 FGJH   96 10 28 1979
Lovelace, Ada              183099831 BAMS    0  6 30 1992
Babbage, Charles           419309823         0  8 31 1892
```

Input File: bart.in
Output File: bart.out
Source File: bart.<xxx>

The Bart Challenge

Problem Description

After making a purchase at the comic book store, Bart Simpson change was 17 cents. He received 1 dime, 1 nickel, and 2 pennies. Later that day, He was shopping at a convenience store. Again his change was 17 cents. This time he received 2 nickels and 7 pennies. He began to wonder: "How many stores can I shop in and receive 17 cents change in a different configuration of coins?" After a suitable mental struggle, he decided the answer was 6. He then challenged you to consider the general problem.

Write a program that will determine the number of different combinations of US coins (penny, nickel, dime, quarter, half-dollar) which may be used to produce a given amount of money.

The input will consist of a set of numbers between 0 and 99, inclusive; one per line in the input file.

The output will consist of the appropriate statement from the selection below; on a single line in the output file for each input value. The number *m* is the number your program computes, *n* is the input value.

```
There are m ways to produce n cents change.
There is only 1 way to produce n cents change.
```

Sample Input

```
17
11
4
```

Sample Output

```
There are 6 ways to produce 17 cents change.
There are 4 ways to produce 11 cents change.
There is only 1 way to produce 4 cents change.
```

Input File: puzzle.in
Output file: puzzle.out
Source File: puzzle.<xxx>

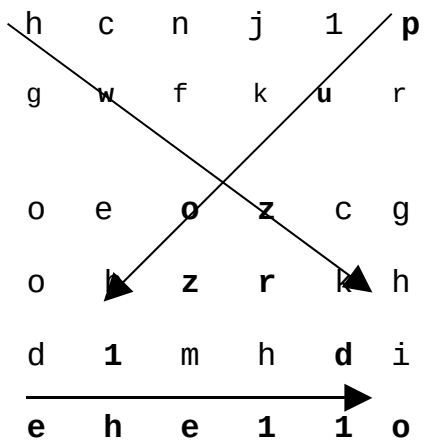
Word Puzzle

Puzzle Description

In a word puzzle you are given a list of words and a grid of letters. The objective of the game is to find all listed words embedded in the grid of letters. The words can be found horizontally, vertically or diagonally.

Make a program that will solve a word puzzle. Your program must find all words present and output the position of the first and last letters of the word on the grid. The input file consists of the dimension of the grid, the grid of letters and a list of words to find.

The output file will be the list of letters, followed by the starting coordinate and ending coordinate, or the string "not found".



The following figure depicts the grid used in the example below.

Sample Input

```
6
ACNJLP

GWFKUR

OEOZCG
```

```
OBZRKH
DLMHDI
EHELLO
puzzle
word
problem
hello
array
input
output
```

Sample Output

```
puzzle (0, 5) (5, O)
word     (1, 1) (4, 4)
problem not found
hello     (5, 1) (5, 5)
array  not found
input  not found
output  not found
```

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '98

Expert Division

Sponsored by

***AEIC* and Lucent Technologies**

Contents

Input File: code.in
Output File: code.out
Source File: code.[cpf]

Code Generator

Problem Description

Your employer needs a backend translator for a very SIC (Simplified Instructional Computer) machine. Input to the translator will be arithmetic expressions in postfix form and the output will be SIC assembly language code.

The target machine has a single register and the following instructions, where the <operand> is either an identifier or a storage location.

load	<operand>	-- Loads <operand> into register
add	<operand>	-- Add <operand> to register contents
sub	<operand>	-- Subtracts <operand> from register contents
mult	<operand>	-- Multiplies register contents by <operand>
div	<operand>	-- Divides register contents by <operand>
neg		-- Negates register contents
str	<operand>	-- Stores register contents at address <operand>

An arithmetic operation replaces the contents of the register with the expression result. Temporary storage locations are allocated by the assembler for an operand of the form "$n" where n is a single digit.

The input consists of several legitimate postfix expressions, each on a separate line. Expression operands are single letters and operators are the normal arithmetic operators (+,-,*,/) and unary negation(@).

The output must be assembly language code that meets the following requirements:

1. One instruction per line, with the instruction mnemonic separated from the operand (if any) by one blank.
2. One blank line must separate the assembly code for successive expressions.
3. The original order of operands must be preserved in the assembly code.
4. Asse*mbly code for each operator must be generated as soon as it is encountered.

5. As few temporaries as possible should be used(given the above restrictions).
6. For each operator in the expression, the minimum number of instructions must be generated(given the above restrictions).
7. The last instruction will leave the result of the expression on the register.

Sample Input

```
AB+CD+EF++GH+++
AB+CD+-
AB/CD//
```

Sample Output

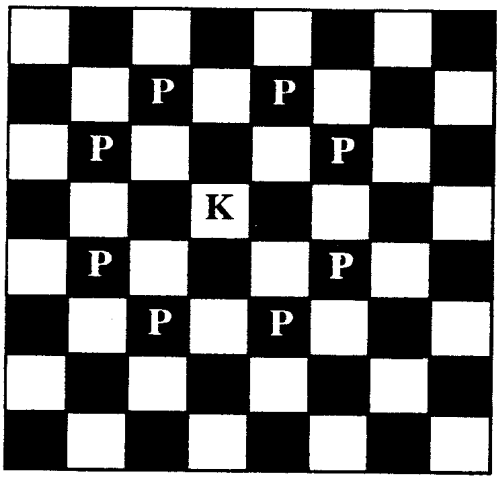
```
load A
add B
str $1
load C
add D
str $2
load E
add F
add $2
str $2
load G
add H
add $2
add $1

load A
add B
str $1
load C
add D
neg
add $1

load A
div B
str $1
load C
div D
str $2
load $1
div $2
```

Input File: knight.in
Output File: knight.out
Source File: knight.[xxx]

Knight Tour

Problem Description

On chess, the knight piece has one of the most complicated moves. You can think of the Knight's move as an "L". It moves two squares either horizontally or vertically and then makes a right angle turn for one more square. It can also move one square either horizontally or vertically and then make a right angle turn for two more squares. In the figure below, the knight (**K**), can move to any of the squares occupied by the pawns(**P**).



The objetive of this problem is, given a starting coordinate for the knight, to move it through all the squares in the board, visiting each square only once. A square is identified by its coordinates. For example the square on the top right corner is (1,1). The one next to it is (1,2). Output from the program should be a sequence of 64 coordinates. There will be eight coordinates per line. Each coordinate separated from the next by a blank space.

Notes:

- There is more than one correct answer for each input. Your program must print only one
- Note that brute force will not achieve a correct answer in a reasonable time. You should use the knowledge about the movement of the Knight to limit the search. Your solution should attain a correct answer in a reasonable time.

Sample Input

1 1

Sample Output

```
1  1
2  3
3  1
1  2
2  4
1  6
2  8
4  7
6  8
8  7
7  5
8  3
7  1
5  2
7  3
8  1
6  2
4  1
2  2
1  4
2  6
1  8
3  7
5  8
7  7
8  5
6  6
7  8
8  6
7  4
8  2
6  1
4  2
2  1
1  3
3  2
5  1
4  3
3  5
5  4
3  3
2  5
1  7
3  8
5  7
4  5
6  4
```

5	6
4	4
6	3
8	4
7	2
5	3
6	5
4	6
3	4
1	5
2	7
4	8
3	6
5	5
6	7
8	8
7	6

Input File: dna.in

Output File: dna.out
Source File: dna.[xxx]

DNA Translation

Problem Description

Deoxyribonucleic acid (DNA) is composed of a sequence of nucleotide bases paired together to form a double-stranded helix structure. Through a series of complex biochemical processes the nucleotide sequences in an organism's DNA are translated into proteins it requires for life. The object of this problem is to write a computer program which accepts a DNA strand and reports the protein generated, if any, from the DNA strand.

The nucleotide bases from which DNA is built are adenine, cytosine, guanine and thymine (hereafter referred to as A, C, G and T, respectively). These bases bond together in a chain to form half of a DNA strand. The other half of the DNA strand is a similar chain, but each nucleotide is replaced by its complementary base. The bases A and T are complementary, as are the bases C and G. These two "half-strands" of DNA are then bonded by pairing of the complementary bases to form a strand of DNA.

Typically a DNA strand is listed by simply writing down the bases which form the primary strand (the complementary strand can always be created by writing the complements of the bases in the primary strand). For example, the sequence TACTCGTAATTCACT represents a DNA strand whose complement would be ATGAGCATTAAGTGA. Note that A is always paired with T, and C us always paired with G.

From a primary strand of DNA, a strand of ribonucleic acid (RNA) known as messenger RNA (mRNA for short) is produced in a process known as transcription. The transcribed mRNA is identical to the complementary DNA strand with the exception that thymine is replaced by a nucleotide known as uracil (hereafter referred as ti U). For example, the mRNA strand for the DNA in the previous paragraph would be AUGAGCAUUAAGUGA.

It is the sequence of bases in-the mRNA which determines the protein that will be synthesized. The bases in the mRNA can be viewed as a collection codons, each codon having exactly three bases. The codon AUG marks the start of a protein sequence, and any of the condons UAA, UAG or UGA marks the end of sequence. The one or more condons between the start and termination codons represent the sequence of amino acids to be synthesized to form a protein. For example, the mRNA codon AGC corresponds to the amino acid serine (Ser), AUU corresponds to isoleucine (ILe), and AAG corresponds to lysine (Lys). So, the protein formed from the example mRNA in the previous paragraph is, in its abbreviated form Ser-lLe-Lys.

The complete genetic code from which codons are translated into amino acids is shown in the table below (note that only the amino acid abbreviations are shown). It should also be noted that the sequence AUG, which has already

been identified as the start sequence, can also correspond to the amino acid methionine (Met), So, the first AUG in a mRNA strand is the start sequence, but subsequent AUG codons are translated normally into Met amino acid.

First base in codon	Second base in codon				Third base in codon
	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	---	---	A
	Leu	Ser	---	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gin	Arg	A
	Leu	Pro	Gin	Arg	G
A	lie	Thr	Asn	Ser	U
	lie	Thr	Asn	Ser	C
	lie	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

The input for this program consists of strands of DNA sequences, one strand per line, from which the protein it generates, if any, should be determined and output. The given DNA strand may be either the primary or the complimentary DNA strand, and it may appear in either forward or reverse order, and the start and termination sequences do not necessarily appear at the ends of the strand. For example, a given input DNA strand to form the protein Ser-iLe-Lys could be any of ATACTCGTAATTCACTCC, CCTCACTTAATGCTCATA, TATGAGCATTAAGTGAGG or GGAGTGAATTACGAGTAT. The input file will be determined by a line containing a single asterisk character.

You may assume the input to contain only valid, upper-case, DNA nucleotide base letters (A, C, G and T). No input line will exceed 255 characters in length. There will be no blank lines or spaces in the input. Some sequences, though valid DNA strands, do not produce valid protein sequences; the string "*** No translatable DNA found ***" should be output when an input DNA strand does not translate into a valid protein.

Sample Input

ATACTCGTAATTCACTCC

CACCTGTACACAGAGGTAACTTAG

Sample Output

Ser-Ile-Lys

Cys-Leu-His

ICOM Challenge Programming Contest
March 28, 1998 Expert Division

Input File: calc.in
Output File: calc.out
. Source File: calc.[cpf]

The Etruscan Calculator

Problem Description

The two properties of arithmetic operators which determine the order in which they are executed in an expression are their precedence level and their associativity. Precedence levels determine the order in which different operatom are executed. Associativities determine the order in which a repeated operator is executed. Left associative operators are evaluated left to right while right associative operators are evaluated right to left. Different operators with equal precedence levels are evaluated left to right. For example:

- If the precedence level of * is higher than that of +, the expression 5*3+4 would evaluate to 19.
- If the precedence level of + is higher than that of *, the expression 5' 3 + 4 would evaluate to 3 5.
- Left-associativity of - would cause the expression 3-2-1 to be interpreted as (3-2)-1 which evaluates to o.
- Right-associativity of - would cause 3-2-1 to be interpreted as 3-(2-1) which evaluates to 2.

Archaeologists have unearthed new evidence that indicates the Etruscans, ancient inhabitants of what we now call Italy, had a fairly complex arithmetic system. The operations used where the usual binary operations, but the symbols used, their precedence and their associativities, vary widely across the region. Your team is to write a program that will implement a simple integer Etruscan calculator with operations +, -, *, and /, where / denotes integer division. The catch is, your calculator has to deal with all of the local dialects.

The input file will contain 4 lines that describe the rules of precedence and associativity for each symbol, followed by one or more lines each containing an expression using the new symbol set to be evaluated. The first four lines each contain a four character string $c_1c_2c_3c_4$ beginning in column one, where:

⇒ C_1 denotes the standard operator that is being described

⇒ C_2 denotes the local symbol being used for that operator

⇒ C_3 is a digit denoting the local precedence of the operator (higher digit means higher precedence)

⇒ C_4 is a single letter denoting the Joeal associativity of the operator (L for left associativity, R for right).

The line

-@1R

means that the symbol @ will be used to denote minus which will be right associative and have precedence 1. The expression 5@3@1 under these circumstances will evaluate to 3.

For each input expression, your program must print one line containing the expression with standard operators followed by a space, an equal sign, and the result.

Sample Input

```
+@1L
-+3R
*-2R
//2R
1@1
5@5+4
2@3@1 2/6/5+3
```

Sample Output

1+1=2
5+5-4 = 6
2+3*12/6/5-3= 14

Notes: The last expression, parenthesized to show you the order of execution is (2+((3*12) / (6/(5-3)))).
Although the Etruscan civilization existed, little is known about their arithmetic system and its written form. The characteristics we attribute to their system in this problem are purely fictional.

Input File: vision.in
Output File: vision.out
Source File: vision.[cpf]

Cybervision

Problem Description

Jayuya Robotics Inc., a local robot manufacturer, has asked for your help to develop the software for the automatic recognition feature of their robot vision system. Their robots have cameras that capture images of the working space, and moving arms that can lift and move objects to and from any point inside the working space. Your module must recognize objects on an image grabbed by the camera, and report their position to a higher-level module.

An image can be represented by a 2-dimensional matrix of pixels. The camera captures pictorial information using a 2-dimensional grid of sensors, such that each sensor has a corresponding pixel in the image. Light intensity information gets digitally coded into pixels according to an intensity scale. In a binary image, the scale has only two possible codes: 1 for high intensity and 0 for low intensity.

Jayuya's robots capture binary images only. Objects to be recognized are dark (low intensity) and the background is bright (high intensity), as per manufacturer's specifications.

Your job is to develop a program that takes a binary image, finds any objects in it, and outputs their position (2-dimensional center of mass) relative to the pixel in the upper-left corner of the image. The coordinates for the pixel on the upper-left corner are (0,0). The center of mass *C(x~,yc)* should be computed using the equations:

$$x_c = \frac{\sum_{1}^{np} x_i}{np},$$

$$\text{and } y_c = \frac{\sum_{1}^{np} y_i}{np},$$

where *np* is the number of pixels that belong to the object.

Objects can have any shape and size. The following rules may be used to identify objects:

(1) If a pixel is 0 (dark), then it belongs to an object. If a pixel is I (bright) it belongs to the background.
(2) Pixels on the border of the image cannot be dark.
(3) Objects are formed by one or more connected dark pixels. Two dark pixels are connected if their x and y coordinates differ by at most 1 (i.e. For all $1 < i < n$, $1 < j < m$, where n is the height and m is the width of the image, If pixel $P(i,j)$ is dark, then all dark pixels $P(p,q)$, such that $i-1 \leq p \leq i + 1$ and $j - 1 \leq q \leq j + 1$, belong to the same object).

Hint: First, use the rules above to find out what object each pixel belongs to, and then compute the center of mass for each object using the coordinates of each pixel of that object.

Sample Input

The first line contains the height and width of the image, in that order. This is followed by the matrix of pixels. Each row is in a new line. There are no spaces between column pixels. Your program must handle image up to 50x50 in size.

```
10  20
11111111111111111111
11111111111111111111
11100111001110001111
11100101001100100111
11100000001001110011
11100000001011011011
11111000111001110011
11111101111100100111
11111111111110001111
11111111111111111111
```

Sample Output

```
The input image contains 3 objects.
(6.000,4.111)
(14.000,5.000)
(14.000,5.000)
```

Centers of mass must be sorted by increasing x (column) coordinate. In case there are centers of mass with equal x coordinates, they must be ordered further by increasing y (row) coordinate. Note that the coordinates are rounded to the nearest thousandth.

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '98

Intermediate Division

Sponsored by
AEIC and Lucent Technologies

Contents

Input File: egypt.in

Output File: egypt.out

Source File: egypt.[xxx]

Egyptian Multiplication

Problem Description

In 1858, A. Henry Rhind, a Scottish antiquary, came into posession of a document which is now called the Rhind Papyrus. Titled " Directions for Attaining Knowledge into All Obscure Secrets", the document provides important clues as to how the ancient Egyptians performed arithmetic.

There is no zero in the number system. There are separate characters denoting ones, tens hundreds, thousands, ten-thousands, hundred-thousands, millions and ten-millions. For the purposes of this problem, we use near ASCII equivalents for the symbols.

- I for one
- n for ten
- 9 for hundred
- 8 for thousand
- r for ten-thousand

(For the purpose of this problem we are not considering numbers bigger than 99,999)

Numbers were written as a group of ones preceded in turn by groups of tens, hundreds, thousands and ten-thousands. Thus our number 4, 023 would be rendered: III nn 8888. Notice that a zero digit is indicated by a group consistingof none of the corresponding symbol. The number 40, 230 would be thus be rendered: nnn 99 rrrr.

To multiply two numbers a and b, the Egyptians would work with two columns of numbers. They would begin by writing the number I in the left column beside the number a in the right column. They would proceed to form new rows by doubling the numbers in both columns. Notice that doubling can be affected by copying symbols and normalizing by a carrying process if any group of symbols is larger than 9 in size. Doubling would continue as long as the number in the left column does not exceed the other multiplicand b. The numbers in the first column that summed to the multiplicand b were marked with asterisk. The number in the right column alongside the asterisks were then added to produce the result.

You are to write a program to perform this Egyptian multiplication. Input will consist of several pairs of nonzero numbers written in Egyptian system described above. There will be one number per line; each number will consist of groups of symbols, and each group is terminated by a single space (including the last group).

For each pair of numbers, your program should print the steps described above used in Egyptian multiplication. Numbers in the left column should be in line with the left margin. Each number in the left and right column will be represented by groups of symbols, and each group is terminated by a single space (including the last group). If there is an asterisk in the left column, it should be separated from the end of the left number by a single space. Up to the 40th character position should then be filled with spaces. Numbers in the right column should begin at the 41st character position on the line and end with a newline character.

Test data will be chosen to ensure that no overlap can occur between columns. After showing each of the doubling steps, your program should print the string: " The solution is: " followed by the product of the two numbers in Egyptian notation.

Below we show the steps corresponding to the multiplication of 483 by 27.

Sample Input

```
III nnnnnnnn 9999
IIIIIII nn
```

Sample Output

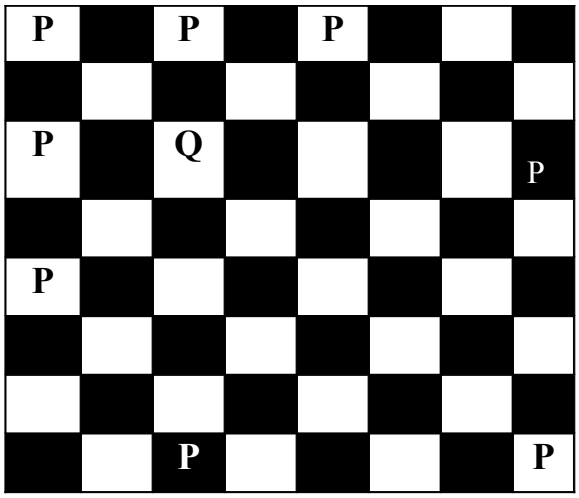
```
I *                                III nnnnnnnn 9999
II *                               IIIIII nnnnnn 999999999
IIII                               II nnn 999999999 8
IIIIIIII *                         IIII nnnnnn 99999999 888
IIIIII n *                         IIIIIIII nn 9999999 8888888
The solution is: I nnnn 888 r
```

Input File: queen.in
Output File: queen.out
Source File: queen.[xxx]

Queen Tour

Problem Description

In chess, the queen is the most powerful pice on the board. It can attack any piece that is located horizontally, vertically or diagonally, independent of the number of squares form her. In the picture bellow, all the pawns (P) are under attack by the queen (Q).



The objective of this problem is to place 8 queens in a chessboard in such a way that none of the queens are under attack by each other. Your program will read the position of one queen, and output the positions of the rest of the queens as shown below. The are no restrictions on where to place any of each input. Just output one of the solutions. Left corner is (1,1). You can assume there is at least one valid arrangement given the position of the first queen.

Notes:

- Note that brute force may not achieve a correct answer in a reasonable time. You should use the knowledge about the movement of the queen to limit the search. Your solution should attain a correct answer in a reasonable time.

Sample Input

```
1 2
```

Sample Output

```
1 2
2 5
3 7
4 1
5 3
6 8
7 6
8 4
```

Input File: sidewlk.in
Output File: sidewlk.out
Source File: sidewlk.[xxx]

On the Sidewalk

Problem Description

Consider an array of floor tiles covering a sidewalk. Every floor tile is numbered from 0 to N in ascending order. N is a positive non-zero integer smaller than 40.

There are children playing on the sidewalk. Children start to play at tile number 0. Children move forward in steps of either one or two tiles. There are no backward moves allowed. All children continue to move until they reach tile N, where they stop and wait for the rest of the children to arrive.

The path followed by a child may be represented by a sequence of tile numbers, resembling the tiles the child stopped at. Given N, your program will calculate the number of possible paths a child followed to reach tile N.

The input file consists of a series of integers, one number per line. Each number is a new value for N for which your program will output the correct answer, as shown below.

Sample Input

1
3
7
8

Sample Output

1
3
21
34

ICOM Challenge Programming Contest
March 28, 1998
Intermediate Division

Input File: life.in
Output File: life.out
Source File: life.[xxx]

John Conway's Game of Life

Problem Description

The Game of Life was invented by John Conway. The game is played on a field of cells, each of which has eight neighbors (adjacent cells). A cell is either occupied (by an organism) or not. The rules for deriving a generation from the previous one are these:

Death:
If an occupied cell has 0,1,4,5,6,7, or 8 occupied neighbors, the organism dies (0,1: of loneliness; 4 thru 8: of overcrowding).

Survival:
If an occupied cell has two or three neighbors, the organism survives to the next generation.

Birth:
If an unoccupied cell has exactly three occupied neighbors, it becomes occupied.

Your program is to simulate the game of life on a 15 x 15 grid for a given number of generations, using an input grid as generation 0. The input file will consist of the number of generations to be displayed and a list of coordinates for organisms in generation 0. The coordinates will be of the form 'x y' where x is the row and y is the column. The range for x and y will be from 0 to 14. The upper-left corner is (0,0). You may assume proper coordinates will be input.

Your program must output a grid for each gener~.tion(including generation 0), preceded by a line for the name of the generation, as shown below. An empty cell will be represented in the output by '-', and a living cell will be represented by 'x'.

See sample input and output on the next page.

Sample Input

```
4
6 0
6 1
6 2
6 3
5 4
7 4
```

Sample Output

```
Generation 0
---------------
---------------
---------------
---------------
---------------
----x----------
xxxx-----------
----x----------
---------------
---------------
---------------
---------------
---------------
---------------
---------------
Generation 1
---------------
---------------
---------------
---------------
---------------
-xxx-----------
-xxxx----------
-xxx-----------
```

```
---------------
---------------
---------------
---------------
---------------
---------------
---------------
```

Generation 2

```
---------------
---------------
---------------
---------------
--x------------
-x--x----------
x---x----------
-x--x----------
--x------------
---------------
---------------
---------------
---------------
---------------
---------------
```

Generation 3

```
---------------
---------------
---------------
---------------
---------------
-x-x-----------
xx-xxx---------
-x-x-----------
```


Generation 4

----
xx-x----------
xx-x----------
xx-x----------

Input File: galac.in
Output File: galac.out
Source File: galac.[xxx]

Galactic Import

Problem Description

With the introduction of the new ThrustoZoom gigadimensional driver, it has become possible for HyperCommodities, the import/export conglomerate from Chicago, to begin trading with even the most remote galaxies in the universe. HyperCommodities wants to import goods from some of the galaxies in the Plural Z sector. Planets within these galaxies export valuable products and raw materials like vacuuseal, transparent aluminium, digraphite and quantum steel. Preliminary reports have revealed the following facts:

- *Each galaxy contains at least one and at most 26 planets. Each planet within a galaxy is identified by a unique letter from A to Z.*
- *Each planet specializes in the production and export of one good. Different planets within the same galaxy export disfferent goods.*
- *Some pairs of planets are connected by hyperspace shipping lines, If planets A and B are connected, they can trade goods freely. If planet C is connected to B but not to A, then A and C can still trade goods with each other through B, but B keeps 5 % of the shipment as a shipping fee. (Thus A only receives 95 % of what C shipped, and C receives only 95 % of what A shipped.) In general, any two planets can trade goods as long as they are connected by some set of shipping lines, but each intermediate planet along the shipping route keeps 5 % of what it shipped (which is not necessarily equal to 5% of the original shipment).*
- *At least one planet in each galaxy is willing to open a ThrustoZoom shipping line to Earth. A ThrustoZoom line is the same as any other shipping line within the galaxy, as far as business is concerned. For example, if planet K opens a ThrustoZoom line to Earth, then the Earth can trade goods freely with K, or it can trade goods with any planet connected to K, subect to usual shipping fees.*

HyperCommodities has assigned a relative value (a positive real number less than 10) to each planet's chief export. The higher the number, the more valuable the product. More valuable products can be resold with a higher profit margin in domestic markets. The problem is to determine which planet has the most valuable export when shipping fees are taken into account.

The input consists of one or more galaxy descriptions. Each galaxy description begins with a line containing an integer N which specifies the number of planets in the galaxy. The next *N* lines contain descriptions of each planet, which consist of:

1. The letter use to represent the planet.
2. A space
3. The relative value of the planet's export, in the form d.dd.
4. A space
5. A string containing letters and/or character * a letter indicates a shipping line to that planet, and a ' indicates a willingness to open a ThrustoZoom shipping line to Earth.

For each galaxy description, output a single line which reads "Import from P" where P is the letter of the planet with the most valuable export, once shipping fees have been taken into account. (If more than one planet have the same most valuable export value then output the planet which is alphabetically first.)

Sample Input

```
1
F 0.81 *
5
E 0.01 *A
D 0.01 A*
C 0.01 *A
A 1.00 EDCB
B 0.01 A*
10
S 2.23 Q*
A 9.76 C
K 5.88 MI
E 7.54 GC
M 5.01 OK
G 7.43 IE
I 6.09 KG
C 8.42 EA
O 4.55 QM
Q 3.21 SO
```

Sample Output

```
Import from F
Import from A
Import from A
```

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '98

Beginner Division

Sponsored by

***AEIC* and Lucent Technologies**

Contents

Output Fire: cornbin~out
Soume FiLe: connbin.EXXX

Combinations

Problem Description

Giver an array of integers A ard a separate integer N. determine if there is a subset of A such that the sum of its elements is equal to N.

The input will have the number N in a separate line, followed by the elements of A. each in a separate line The end of the array will be marked by the end of file. The maxt~um size of the array is 100.

The first line of a valid output should state if the result was positive or negative (i.e. "There is at least ono valid combination!." or No valid combination was fcund.~). If a valid combination is found print the numbers that are part of it

Sample Input

1023
1
2
4
8
12
16
32
63
64
128
256
512
1024
2048

Sample Output

There is at least one valic zom>ir!c~toflL
63 64 128 256 515

ICOM Challenge Programming Contest
March28, 1998
Beginner Division

Input File:maya.in
Output File: maya.out
Source File: maya.[xxx]

Maya Calendar

Problem Description

During his last sabbatical, professor M.A .Ya made Surprising discovery about old Maya calendar. From and old knotted message, professor discover that the Maya civilization used a 365 .day long year, called Haab , which has 19 months. Each of the first 18 months was 20 days long , and the names of the months were pop no , zip , zotz , tzec , xul , yoxkin , mol , chen yax , zac ceh mac , kannin , muan, pax , koyab, cumhu. Instead of having names , the days in the months were denoted by numbers starting , from 0 to 19. The last month of Haab was called uayet and had 5 days denoted by numbers O, 1, 2, 3, The Maya believed that this month was unlucky, the courtof justice was not in session, the trade stopped, people did not even sweep the floor.

For religious purposes, the Maya used another calendar in which the year was called Tzolkin (holly year). The year was divided into:thirteen periods, each*20* days long. Each day was denoted by a pair consisting of a number and the name of the day. They used 20 names: imix , ik , akbal , kan , chicchan , cimi,
manik, lamat, muluk , ok, chuen, eb , ben, ix, mem, cib
caban eznab, canac , ahau , and 13 numbers; both in cycles.

Notice that each day has an ambi description. For example, at the beginning of the year the days were described as follows:

1 imix, 2 ik, 3 akbal, 4 kan, 5 chicchan, 6 cimi, 7 manik , 8 lamat , 9 muluk , 10 ok , 11 chuen, 12 eb, 13 ben,1 ix, 2, mem, 3 cib, 4 caban , 5 eznab , 6 canac , 7 ahau ,and again in the next period 8 imix D ik , 10 akbal ...

Years (both Haab and Tzolkin) were denoted by numbers O, 1…….the number 0 was the beginning of the world. Thus the first day was:

Haab: O. pop 0

Tzolkin: 1 imixO

Help professor M.A. Ya and write a program for him to convert the dates from the Haab calendar to the Tzolkin calendar.

Sample Input

The date in Haab is given in the following format:

NurnberOfTheDay. Month Year:

The first line in the file contains the number of the input dates in the file. The next n lines contain n dates in the Haab calendar format, each in a separate line. The year is smaller than 5000.

```
1
10. zac 0
```

Sample Output

The date in Tzolkin should be in the following format:

Number NameOfDay Year
The first line of the output file contains the number of the output dates. In the next n lines, there are dates in the Tzoikin calendar format, in the order corresponding to the input dates.

```
1
3 chuen 0
```

Input File: pall.in
Output File: pall.out
Source File: pali.[xxx]

Palindrome Detection Using Recursion

Problem Description

The goal of this problem is to write a program that will accept a string with a maximum length of 50 and determine if it is a palindrome. A palindrome is a word or string which is spelled the same forwards and backwards. The function you create to solve this problem must be recursive.

Your solution to the palindrome problem must have only one call in the main() function and it must call itself successively thereafter, until an answer is returned. Furthermore your entire program should only take into account numbers and letters when determining if a string is a palindrome, but it should print out the entire string as entered when displaying the answer. The program should not be case sensitive; 'A' and 'a' are equivalent.

Sample Input

A data file with several strings, one string per tine, each string no longer than 50 characters.

```
Abba 8 ab’BA
Abbcb Bb
```

Sample Output

A file displaying the string and whether or not the string is considered a palindrome.

"Abba 8 ab’BA" is a palindrome.
"Abbcb Bb" is not a palindrome.

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '98

Beginner Division

Sponsored by

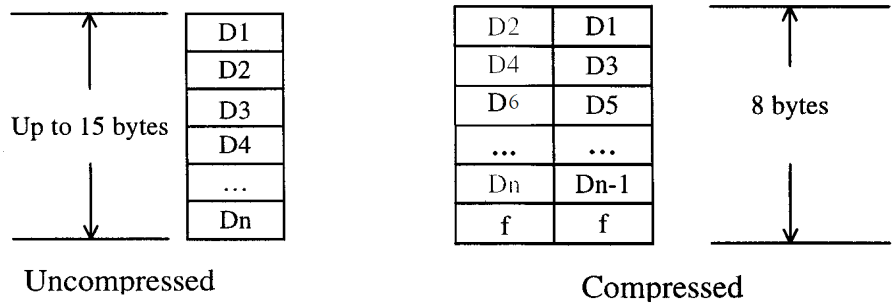
AEIC and Lucent Technologies

ICOM Challenge Programming Contest
March 28, 1998
Beginner Division
Input File: comp.in
Output File: comp.out
Source File: comp.[xxx]

Compression

Problem Description

In Telephony applications, we often receive messages which contain a stream of digits to be manipulated, stored, or otherwise passed to other applications. In a particular example, we can receive up to 15 digits, with values ranging from 0 to 9, inclusive. These digits arrive to our system in a stream of bytes characters ,one digit per byte. Our system needs to store these digits into an existing structure; however, we only have 8 bytes of free space available. Write a program that compresses up to 15 digits into an array of 8 bytes. The compressed array should contain 2 digits per byte. The following is a memory layout of the arrays.



Note that empty slots in the compressed structure contain the hexadecimal value f. This helps determine the end of the stored digits.

Input

The input file contains rows of digits separated a by spaces.

```
5551212
121
```

Output

The output of your program should contain:

· The number of digits in the row.

- The digits on the uncompressed stream represented as a two digit Hexadecimal value up to the number of digits in the row.
- The entire compressed structure (including empty slots)
- Separate each stream output a blank line

7
05050501020102
551512f2ffffffff

3
010201
21 f 1 fffffffffff

ICOM Challenge Programming Contest
March 28, 1998
Beginner Division

Input File: sort.in
Output File: sort.out
Source File: sort.[xxx]

Sorting Problem

Problem Description

Given two arrays A and B of sorted positive integers:

- Every element in the array is greater or equal to 0
- Number of elements in each array < 25
- One or more occurred of the same number arrays.

Code a program that produce a sorted list of elements after combining the elements of array A and B without using a third array or list to perform the sorting and without repeating numbers in the output.

Sample Input

- Two lines of positive integers: first line corresponds to array A, second one corresponds to array B.
- Each array element is separated by space.

1 4 5 7 8 32 45 57 65 67 78
0 4 7 8 11 15 21 27 34 45 57

Sample Output

Sorted list of elements

0 1 4 5 7 8 11 15 21 27 32 34 45 57 65 67 78

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '97

Expert Division

Sponsored by

AEIC and Lucent Technologies

ICOM Challenge 1997 Programming Contest
March 15, 1997

EXPERT DIVISION

Problem: HTML Tables

Write a program that given an HTML source code displays the table included in the document. The display should include borders and the table's content.

HTML (HyperText Markup Language) is a markup language which consists of tags embedded in the text of a document. The browser reading the document interprets these markup tags to help format the document for subsequent display to a reader.

- ·A table is created using the <TABLE> markup tag. The simplest table consists of a single data cell.
    - ·BORDER
        - Specifies that a border is to be placed around the table cells. The width of the border is optionally specified with BORDER--n.
- The markup <TD> defines the start and </TD> defines the termination of a table data cell.
- To form a table of many rows, the markup tag <TR> is inserted where each new row in the table starts. The termination tag for the row is </TR>.
- The tag <TH> may be used instead of <TD> if the cell is a header to a column of cells.

Example:

The source code for a table looks like:

```
<TABLE BORDER=I>
<TR>
<TH>Name</TH>
<TH>Telephone Number</TH>
 </TR>
<TR>
<TD>Juan</TD>
<TD>888-9999</TD>
</TR>
<TR>
<TD>Maria</TD>
 <TD>999-8888</TD>
</TR>
</TABLE>
```

The table will look like:

Name	**Telephone Number**
Juan	888-9999
Maria	999-8888

The source code for this program is example.html.

Problem- AEIC Words

Design a word processor that reads a file and allows the following commands:

col N - sets the number of columns to display data.
replace <word>-<newword>- replaces all occurrences of <word>
with <newword>.
Erase <word>-erases all occurrences of <word>.
save <filename>- save changes to disk. If no argument is
given, changes will be saved to the same input file.
load <filename>- opens the file.
Quit- exits tke program

Note: By default the word processor will display the data in 40 columns. It is not case sensitive. If a word exceeds the limit of columns it is moved to the next line. The commands are not case sensitive.

The input file will be called filename.txt.

Sample run:

..... FILENAME.TXT .....
This book provides a comprehensive and unified introduction to operating systems. The book emphasizes both fundamental principles and design .issues in contemporary systems. Thus it is both a basic reference and up-to-date survey of the state of the art.

cmd:> load filename.txt

..... FILENAME.TXT .....
This book provides a comprehensive and unified introduction to operating systems. The book emphasizes both fundamental principles and design issues in contemporary systems. Thus it is both a basic reference and up-to-date survey of the state of the art.

cmd:> col 67

..... FILENAME.TXT .....
This book provides a dull and unified introduction to operating systems. The book emphasizes both fundamental principles and design issues in contemporary systems. Thus it is both a basic reference and up-to-date survey of the state of the art.

cmd:> replace comprehensive dull

..... FI[.ENAME.TXT .....
This provides a dull and unified introduction to operating systems. The emphasizes both fundamental principles and design issues in contemporary systems. Thus it is both a basic reference and up-to-date survey of the state of the art.

crnd:> erase book

..... OS.TXT .....
This provides a dull and unified introduction to operating systems. The emphasizes both fundamental principles and design issues in contemporary systems. Thus it is both a basic reference and up-to-date survey of the state of the art.

crud:> save oS.txt

Problem**: Navigation of a Simple Maze**

The goal of the program is given a start location within the maze, to find the exit point and print out the optimal solution. Before we start let's look at a simple maze and transform it to a tree structure.

```
+ - -   + - -   + - -   + - -   + - -   +
|  06      05      04      03   |  06   |
+       + - -   + - -   +       +       +
|  07      08      09   |  02   |  14   |
+       + - -   +       +       +       +
|  11      12   |  10   |  01      13   |
+ - -   + - -   +       +       + - -   +
                    OUT     IN
```

We have labeled each position within the maze with numeric labels to aid the transformation to a tree structure. The transformafion yields:

```
                    (01)
                     /\
                  (13) (02)
                   /     \
                 (14)    (03)
                  /        \
A Dead End      (15)      (04)
                            \
                           (05)
                             \
                            (06)
                              \
                             (07)
                             /  \
                           (11)(08)
                             /    \
        A Dead End        (12)   (09)
                                   \
                                  (10)        Out
```

Searching such a maze reduces to viewing the maze as a tree structure and using an appropriate algorithm. Such an algorithm is the preorder search. The preorder search "travels" around a tree structure, always taking the left branch if possible, and "looks" at each node as it is encountered. In our situation, we are looking for the OUT node. Given the ma~e below; write a program to implement the preorder algorithm (hint: it's recursive) to find the solution to the maze.

```
                    Column
Row      0 1 2 3 4 5 6 7 8 9 0 1
        +-+-+-+-+-+-+-+-+-+-+-+-+
      0 |                       |
        + +-+-+ +-+-+-+-+-+-+ + +
      1 | | | |               | | |
        + + + + +-+ + + + + + + +
      2 |       |   | | | | | | |
        + + + + +-+-+ + + + + + +
      3 | | | |   |   | | | | | |
        + + + +-+ +-+ + + + + + +
      4 | | |   | |   | | | | | |
        + + +-+-+ + +-+ +-+ + + +
      5 | | | |   |   | | | | | |
        + +-+-+-+ + +-+ + + + +_|
      6 | |     | | | | | | | | |
        + +-+-+-+ + +-+ + + +-+ +
      7 |       | | |     |     |
        +-+-+-+-+-+ +-+-+-+-+-+-+
                    Out
```

Your program should prompt for starting coordinates, search the maze, then print the solution. A sample run might look like this:

THE AMAZING MAZE PROGRAM

```
                    Column
Row      0 1 2 3 4 5 6 7 8 9 0 1
        +-+-+-+-+-+-+-+-+-+-+-+-+
      0 |                       |
        + +-+-+ +-+-+-+-+-+-+ + +
      1 | | | |               | | |
        + + + + +-+ + + + + + + +
      2 |       |   | | | | | | |
        + + + + +-+-+ + + + + + +
      3 | | | |   |   | | | | | |
        + + + +-+ +-+ + + + + + +
      4 | | |   | |   | | | | | |
        + + +-+-+ + +-+ +-+ + + +
      5 | | | |   |   | | | | | |
        + +-+-+-+ + +-+ + + + +_|
      6 | |     | | | | | | | | |
        + +-+-+-+ + +-+ + + +-+ +
      7 |       | | |     |     |
        +-+-+-+-+-+ +-+-+-+-+-+-+
                    Out

Enter starting row position: 7
Enter starting column position: 3
```

The solution to the maze is:

The solution to the maze is:

```
                    Column
Row     0 1 2 3 4 5 6 7 8 9 0 1
        +-+-+-+-+-+-+-+-+-+-+-+
    0   |* * * *               |
        + +-+-+ +-+-+-+-+-+-+ + +
    1   |*| | |* * * *       | | |
        + + + + +-+ + + + + + + +
    2   |*       |    |*| | | | | |
        + + + + +-+-+ + + + + + +
    3   |*| | |    |   *| | | | | |
        + + + +-+ +-+ + + + + + +
    4   |*| |    | |* *| | | | | |
        + + +-+-+ + +-+ +-+ + + +
    5   |*| | |    |*   | | | | | |
        + +-+-+-+ + +-+ + + + + |
    6   |*|      | |*| | | | | | |
        + +-+-+-+ + +-+ + + +-+ +
    7   |* * * S| |*|      |      |
        +-+-+-+-+-+ +-+-+-+-+-+-+
                    Out
```

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '97

Intermediate Division

Sponsored by

AEIC and Lucent Technologies

INTERMEDIATE DIVISION

Problem: Word Morphing

In this problem you will transform words into other words, one character at a time.

PROBLEM STATEMENT:

1. 1-morphs: The set of 1-morphs of a word is the set of words in which one letter is changed and o letters are rearranged. Write the program ehich prints a list os all the 1-morphs of a word. You will be provided with a dictonary file named WORDS of acceptable words in alphabetical order, one word to a line.
2. Laddergrams: A laddergramsis a chain of 1-morphs which transform a stating word into an ending word. For example, the shortest laddergram changing the word "lead" into the word "gold" is lead load goad gold. Write a program which finds and prints a laddergram of shortest length between the supplied starting and ending words (using the same supplied dictionary).

TECHNICAL CONSTRAINS

The dictionary file is dict.in.

SAMPLE RUNS:

Part 1:

The 1-morphs of TIMES are:
Timer timed tires tiles tides tames limes dimes

Part 2:

Here are some shortest solutions you can use to test your program:

LOVE to HATE: love lone lane late hate
BRAIN to THINK: brain train trait tract track trick thick think
WHITE to BLACK: white whine shine spine spice slice slick slack black

Problem: **Cryptarithmetic**

Write a program that solves a cryptarithmetic problem. In cryptarithmetic problems, letters stand for digits and the aim is to find a substitution of digits for letter such that the resulting sum is arithmetically correct.

TECHNICAL CONSTRAINS:

1. Each letter must stand for a different digit.

SAMPLE RUNS:

```
 FORTY
  + TEN
  + TEN
-----------
 SIXTY
```

Solution:

```
 29786
    850
    850
----------
 31486
```

F=2
0=9
R=7
T=8
Y=6
E=5
N=0

Problem**: Navigation of a Simple Maze**

The goal of the program is given a start location within the maze, to find the exit point and print out the optimal solution. Before we start let's look at a simple maze and transform it to a tree structure.

```
+ - -   + - -   + - -   + - -   + - -   +
|  06      05      04      03   |  06   |
+       + - -   + - -   +       +       +
|  07      08      09   |  02   |  14   |
+       + - -   +       +       +       +
|  11      12   |  10   |  01      13   |
+ - -   + - -   +       +       + - -   +
                    OUT     IN
```

We have labeled each position within the maze with numeric labels to aid the transformation to a tree structure. The transformafion yields:

```
                    (01)
                     /\
                  (13) (02)
                   /     \
                 (14)    (03)
                 /         \
A Dead End     (15)       (04)
                            \
                           (05)
                             \
                            (06)
                              \
                             (07)
                             /  \
                           (11)(08)
                             /    \
          A Dead End      (12)   (09)
                                   \
                                  (10)        Out
```

Searching such a maze reduces to viewing the maze as a tree structure and using an appropriate algorithm. Such an algorithm is the preorder search. The preorder search "travels" around a tree structure, always taking the left branch if possible, and "looks" at each node as it is encountered. In our situation, we are looking for the OUT node. Given the ma~e below; write a program to implement the preorder algorithm (hint: it's recursive) to find the solution to the maze.

```
                    Column
Row     0 1 2 3 4 5 6 7 8 9 0 1
        +-+-+-+-+-+-+-+-+-+-+-+-+
    0   |                       |
        + +-+-+ +-+-+-+-+-+-+-+ + +
    1   | | | |               | | |
        + + + + +-+ + + + + + + + +
    2   |       |   | | | | | | | |
        + + + + +-+-+ + + + + + + +
    3   | | | |   |   | | | | | | |
        + + + +-+ +-+ + + + + + + +
    4   | | |   | |   | | | | | | |
        + + +-+-+ + +-+ +-+ + + + +
    5   | | | |   |   | | | | | | |
        + +-+-+-+ + +-+ + + + + +_|
    6   | |     | | | | | | | | | |
        + +-+-+-+ + +-+ + + +-+ +
    7   |       | | |     |     |
        +-+-+-+-+-+ +-+-+-+-+-+-+
                    Out
```

Your program should prompt for starting coordinates, search the maze, then print the solution. A sample run might look like this:

THE AMAZING MAZE PROGRAM

```
                    Column
Row     0 1 2 3 4 5 6 7 8 9 0 1
        +-+-+-+-+-+-+-+-+-+-+-+-+
    0   |                       |
        + +-+-+ +-+-+-+-+-+-+-+ + +
    1   | | | |               | | |
        + + + + +-+ + + + + + + + +
    2   |       |   | | | | | | | |
        + + + + +-+-+ + + + + + + +
    3   | | | |   |   | | | | | | |
        + + + +-+ +-+ + + + + + + +
    4   | | |   | |   | | | | | | |
        + + +-+-+ + +-+ +-+ + + + +
    5   | | | |   |   | | | | | | |
        + +-+-+-+ + +-+ + + + + +_|
    6   | |     | | | | | | | | | |
        + +-+-+-+ + +-+ + + +-+ +
    7   |       | | |     |     |
        +-+-+-+-+-+ +-+-+-+-+-+-+
                    Out

Enter starting row position: 7
Enter starting column position: 3
```

The solution to the maze is:

The solution to the maze is:

```
                    Column
Row     0 1 2 3 4 5 6 7 8 9 0 1
        +-+-+-+-+-+-+-+-+-+-+-+
    0   |*  *  *  *            |
        +  +-+-+-+  +-+-+-+-+-+-+  +  +
    1   |*|  |  |*  *  *  *        |  |  |
        +  +  +  +  +-+  +  +  +  +  +  +  +
    2   |*        |     |*|  |  |  |  |  |
        +  +  +  +  +-+-+-+  +  +  +  +  +  +
    3   |*|  |  |     |   *|  |  |  |  |  |
        +  +  +  +-+  +-+  +  +  +  +  +  +
    4   |*|  |     |  |*  *|  |  |  |  |  |
        +  +  +-+-+-+  +  +-+  +-+  +  +  +
    5   |*|  |  |     |*   |  |  |  |  |  |
        +  +-+-+-+-+  +  +-+  +  +  +  +_|
    6   |*|        |  |*|  |  |  |  |  |  |
        +  +-+-+-+-+  +  +-+  +  +  +-+  +
    7   |*  *  *  S|  |*|        |       |
        +-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+
                    Out
```

Problem: **Datetime**

Write a program that reads a string representing a datetime, in the format yyyymmddhhmmss. Where yyyy is the year, mm is the month, dd is the day, hh is the hour (valid values are 00 to 24), mm are the minutes (00 – 60), and ss are the seconds (00 – 60). The program also reads a string representing a time in the following format: hhmmss. The program must then subtract the time from the datetime and display the result. Your program must handle leap years. A year is a leap year if it is divisible by 4, but not by a 100, or if the year is divisible by 400.

TECHNICAL CONSTRAINS

1. Input and output must be done interactively.
2. The input is one string of length 14 representing the datetime, and one string of length 6 representing a time. Your program should validate that the user entered valid values and display and error "Invalid datetime." Or "Invalid time." If not.

SAMPLE RUN:

datetime: 19950311104033
time: 233015
result: 1995031110118

ICOM Challenge 1997
Programming Contest
March 15, 1997

BEGINNER DIVISION

Problem: Ordered Fractions

Consider the set of all reduced rational numbers between 0 and 1 inclusive with denominators less than or equal to N.

Here is the set when N=5:

Write a program that, given an integer N between 1 and 100 inclusive, prints the fractions in order of increasing magnitude. You should also print the total number of fractions. Print a tab after each fraction so they don't run too far off the screen during judging.

TECHNICAL CONSTRAINS:

1. Programs must reject inputs where N is less that 1 or greater than 100.

SAMPLE RUN

Enter the maximum denominator: 5

0/1 1/5 1/4 1/3 2/5 1/2 3/5 2/3 3/4 4/5 1/1

There were 11 fractions.

Problem: **Magic Number**

Write a program that creates a two-dimensional array (n*n) with numbers 1 through n^2. The number n is entered by command line. The numbers are placed in such a way that its column sums, row sums and diagonal sums are equal. The number n must be odd.

TECHNICAL CONSTRAINS:

1. The program must reject inputs where n is even.

SAMPLE RUN:

Enter magic number n: 3

3 is the dimension of the n*n matrix and is an odd number.

8	1	6
3	5	7
4	9	2

Problem: **A "Talk" Packet Sniffer**

Sutano's conversation in "talk" was recorded in a log file by the System Administrator. The log files consists of a series of integers and separated by white spaces or carriage returns. The file has the following format:
Message-type Data-type Datasize Data...

Message type: 0 for received message 1 for sent message

Datatype: 0 indicates that the data field consists of ASCII values 1 indicates that the data field consists of integers

Datasize: Nmber of integers to be read.

Data: the message in the form of integers

Given the log file, decode the message and write on the screen with the following format:

(sent): messagel
(received) :message2
.

Problem: **Dictionary**

Given a text input file build a database that contains the words used in the text file. You can't repeat words and the output file must be in alphabetical order.

Source file = Dict.txt

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '95

Expert Division

Sponsored by

AEIC and Lucent Technologies

ICOM Challenge 1995
Programming Contest
March 11, 1995

EXPERT DIVISION

Problem1. Datetime

Source File Name ED I.XXX

Problem:

Write a program that reads a string representing a datetime, In the format yygymmddhhmmss, where YYYg Is the year, mm is the month, dd is the day, hh is the hour (ualid values are 88 t8 24), mm are the minutes (88 - 69), and ss are the seconds {98 - 69). The program also reads a string representing a time in the following format: hhmmss. The program must then subtract the time from the datetime and display the result. Your program must handle leap years. A year is a leap year if it is divisible by 4, Gui ,et b9 a i 88, or if the year is divisiDie by 4ii6.

Input and output must be done interactively.

Input:

One string of length 14 representing the datetime, and one string of length 6 representing a time. Your program should validate that the user entered valid values and display and error "invalid datetime." or "invalid time." if not.

Example:

Input

datetime: 99583-11
time: 233815

Output:

result: 199503111018

TITULO DEL PROBLEMA

Problem 2. **Christmas Tree**

Source File Name: AD2.XXX

Problem:

Write a program to draw and display a Christmas tree on the screen.

Note: This program will be judged according to how elaborate and Lively your tree is, and not in the time it took. Be creative, impress the judges. Time will be used only in the case of a tie. Of course, the rest of the problems will be judged by time, so if you spend too much time on this one, you will have less time for the others.

Pascal to C Mini-While- Converter

Source File Name: AD3.XXX
Input File Name: AD3.Pas
Output File Name: AD3.C

Define: The while statement in the Pascal language has the following format:

while *boolean-expression* do
program-statement;

ex.

```
while n <= 10 do
    n:=n+x;
```

The while statement in the C language has the following format:

while *(boolean expression)*
program-statement;

ex.

```
while (n <=10)
    n:=n+x;
```

The assignment statement in Pascal has the form

variable := expression;

where expression may be a constant, another variable, or a formula to be evaluated.

In C, the assignment statement has the following form:

variable = expression ;

The begin and end statements In Pascal translate to open ({) and close (}) brackets in C. The end in Pascal is followed by a semicolon (;), but the close bracket in C is not.

The <, >,<--, >- logical operators are the same in both C and Pascal. These are the only ones that your program needs to support. In other words, the Pascal while boolean expressions that your program needs to translate are, for example: n<=18, cnt > 5, count l >= count2, t < 2.

Problem:

Write a Program to translate valid while statements in Pascal to the C language. Your program will read a Pascal while statement from the file AD3.pas, and will write the equiualent C statement into the file AD3.c. Your translator program must consider the case of seueral assignment statements within the single while statement begin and end. You do not haUe to worry about nested while statements. You do not need to worry about arrays either. Rs mentioned abeue, your program only needs to support the <, >, <-, >- which are the same in both C and Pascal.

Sample Input/Output:

Input:

```
while'X>."1 do'
begin
    n:-3;
    j:=n-x;
end;
```

Output:

```
while (x>- I )
{
n-3;
j-n-x;
}
```

Restaurant Database

Source File Name: RD4.XXX

Input/Output File Name: RD4.DB

Problem:

The Departamento de Turismo de Puerto Rico has hired you to write a program that will manage a restaurant database that tourists can use to decide where to go for dinner. The database will contain the following information on restaurants: name, address, phone number, type of restaurant (Chinese, French, Italian, Spanish, etc.), and the tallrig {] Lo 5 *stars).*
The program must display a menu that allows the user to perform the following operations:
(I) add a restaurant to the database
(2) delete a restaurant from the database
(3) update a restaurant's data
(4) find a restaurant ITU the type. If the user wants Chinese restaurants, giue a list of all Chinese restaurants in the database.
(5) exit the program

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '95

Intermediate Division

Sponsored by

AEIC and Lucent Technologies

INTERMEDIATE DIVISION

Problem 1. **Greatest Common Divisor**

Source File name ID1.xxx

Definition:

The greatest common divisor of two integers a and b, GCD(a,b) not both of which are zero, is the largest positive integer that divides both a and B.

The least common multiple of a and b, LCM(a,b) is the smallest nonnegative integer that is a multiple of both a and b and can be calculated using.

$$LCM(a, b) = \frac{/ab/}{GCD(a, b)}$$

Problem:

Write a program that reads two integers, and displays their GCD and the LCM. Input and output shall be done interactively.

Sample Input/ Output:

Input:

1260
198

Output:

GCM = 18
LCM = 13860

Problem 2. **Measurement and Unit Conversion**

Source File Name: ID2.xxx

Problem:

Write a menu-driven program that allows the user the following options:

(1) to convert measurements from minutes to hours
(2) to convert feet to meters (1 foot = 0.3048 meter)
(3) to convert from degrees Fahrenheit to degrees Celsius (F = 1.8C + 32).
(4) To exit the program

Problem 3. Stack Manipulation

Source file name: ID3.xxx

Problem:

Write a program to convert an expression in infix notation to postfix and prefix notation. Input and output shall be done interactively.

Input:

A string no longer than 50 characterrepresenting the expression in infix notation.

Sample Input / Output:

Input:

2 * (3 + 4)

Output:

Postfix notation: 2 3 4 + *
Prefix notation: * 2 + 3 4

Problem 4. Binary to decimal, octal and hex conversion

Source File Name: ID4. XXX

Problem:

Write a program that converts a string of binary digits (0s and 1s) to its decimal, octal and hexadecimal representation. The maximum number of binary digits taken as input is 24.

Sample Input/Output:

Input:

0011010100011010

Output:

Hexadecimal: 351A
Octal: 32432
Decimal: 46362

ICON Challenge 1995

Problem 1 Previous Date

Problem:

W rite a program to determine the date of the previous day for any date given by the user: The number of days in each month is as follows:

Jan - 31	May - 31	Sept. - 30
Feb. - 28 or 29*	June - 30	Oct. - 31
March - 31	July - 31	Nov. - 30
April - 30	Aug. - 31	Dec. – 31

*February has 28 days normally, except in leap years when it has 29 days. A year is a leap year if tit is divisible by 400.

Input:

A string of length 8 representing a date in the form: yyyymmdd where yyy is the year, mm is the month, and dd is the day.

The input must be validated and the error "Invalid Date" must be given if the date entered is not valid.

Output:

A string of length 8 representing the previous date, in the following format: yyyymmdd.

Note:

Input and output must be done interactively.

Sample:

Input:	19950311
Output:	19950310
Input:	19998956
Output:	Invalid Date.

Problem 2. Distance, Midpoint and Slope

Source File Name: BD2.XXX

Definition:

Distance Formula:

The distance d(Pl, P2) between any two points PI(x1, Yl) and P2(x2, Y2) in a coordinate plane is

d(Pl, P2) = (Xl - x2)^2 (Yl - Y2)^2

and P2 from a

Xl + x2 Yl +Y2~

J

2 2

Slope:

Let I be a line that is not parallel to the y axis and let PliXl, Yl) and P2(x2, Y2) be distinct points on I, the slope m of I is:

$$m = \frac{Y2 - Yl}{x2 - xI}$$

If I is parallel to the y axis, then the slope is not defined.

Problem'

Write a program that takes two points P! user, and find the:

(a) distance between the points
(b) the midpoint of the line segment from PI to P2
(c) the slope of the line that passes through PI and P2. If not defined, indicate so.

Input and output must be done interactively.

Sample Input/Output:

Input:

Xl = 5 Yl = 3 x2 = -7 Y2 = 8

Output:

d(P i, P2) = 13.00 midpoint = (-I.00, 5.50) slope = -0.42

Midpoint Formula:

The midpoint of the line segment from P(x, y) to P2(x, y) is:

Problem 3. **Change**

Source File Name: BD3.XXX

Problem:

Write a program that, given all amount of money as input, will return the number of quarters, dimp. s, nickels, and pennies that add up to the amount entered using the minimum number of coins.
Input and output shall be done interactively.

Sample Input/Output:

Input:

2.17

Output:

Q: 8
D: 1
N: 1
P: 2

Problem 4. Text Editing

Source File Name: BD4.XXX

Problem:

Write a program that permits the input of a name consisting of a first name, a middle name or initial, and a last name, in that order , and then prints the last name, followed by a comma , andthen the first and middle initial, each followed by a period. Theinput and output shall be done interactively.

Sample Input/Output:

The input “ John H. Doe “ should produce “ Doe, J. H. “

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '94

Expert Division

Sponsored by

AEIC and Lucent Technologies

EXPERT DIVISION

Problem I. **Memory Management**

Source File Name ED I.XXX

Definition:

The operating system and hardware divide virtual memory into pages of identical size and divide real memory into page frames. During the course of executing a program, the CPU generates a sequence of virtual page references called the reference string:

$$R = r_1 r_2 \ldots r_k$$

If the page r_1 being referenced is resident in a page frame, the reference proceeds normally. Otherwise, if the page is not in memory, a page fault interrupt occurs. When this happens, the CPU scheduler copies the page into memory. If there is no page frame available (free) In memory, the CPU scheduler takes a page from memory and copies it to a disk. It then places the new page in a page frame. This is called swapping.

Problem:

Upon execution of a program, the following reference string is generated:

$$R = 272722(2827229733733)^n37333839322$$

Assume n=I0, which means that the substring within parenthesis is referenced 10 times. Write a program to analyze the run-time behavior if a FIFO mechanism is used by the CPU scheduler to swap a page from memory. That is, the oldest page in memory is taken out of memory when the page being referenced is not in memory and there is no page frame available to copy the new page.
Assume that real memory is empty at the beginning of execution.

In particular, your program should print:

1) IRI - total number of pages referenced.

2) A table with the following Information:

Page frames - the number of page frame in real memory. This should range from I to 15.

Page faults - total number of page faults

Page Fault Rate - average page fault rate (Page Faults/IRI)

Example:

Let R = 223293.

For page frames = 1, this is what goes on during execution:

Page 2 is referenced, it is not in memory, so a page fault occurs. Page 2 is swapped into memory.
Page 2 Is referenced again, it Is in memory.
Page 3 is referenced, it is not in memory and there Is no page frame available (ther is only one page frame and It is taken by
page 2). A page fault occurs. Page 2 is swapped from memory and page 3 is copied into memory.
Page 2 is referenced again, it is not in memory and there is no page frame available. A page fault occurs. Page 3 is swapped from memory and page 2 is copied into memory.
Page 9 is referenced, it is not in memory and there is no page frame available. A page fault occurs. Page 2 is swapped from memory and page 9 is copied into memory.
Page 3 Is referenced, it is not in memory and there is no page frame available. A page fault occurs. Page 9 is swapped from memory and page 3 Is copied into memory.

For page frames = 2:
Page 2 is referenced, it is not in memory, so a page fault occurs. Page 2 Is referenced again, It Is in memory.
Page 3 is referenced, it is not in memory, so a page fault occurs. Page 2 is referenced again, it is in memory.
Page 9 is referenced, it is not in memory and there is no page frame available. A page fault occurs. Page 2 is swapped from memory and page 9 Is copied Into memory.
Page 3 is referenced, it is in memory.

And so on.

Example Output for R- 223293:

The number of pages referenced, IRI, is 6.

Page Frames	Page Faults	Page Fault Rate
1	5	0.8333
2	3	0.50
3	3	0.50
4	3	0.50
.	.	.
.	.	.
.	.	.
15	3	0.50

Turtle Text Graphics

Source File Name: AD2.XXX Input File Name: AD2.1N

MaCrohard Corp has hired you to build a spanky new interpreter called BOGUS. (Beginners lOGo Universal System). BOGUS consists of a ting set of commands used to create extremelU crude turtle text graphics. The turtle is represented by the letter T (of course) and the turtle's trail is represented by dots (.). The interpreter reads the input file and displays the program's outcome on screen. When the turtle hits the end of the screen, the turtle does not wrap around. The turtle sits there until another command takes it back to a position towards the center of the screen or along the edge of the screen. Also, any suntax error must be ignored. Every command occupies one line. Initially, the turtle is in the center of the screen (I 2,48) facing north.

The following are the commands for BOGUS:

I. FWD <steps>
Takes an Integer X and moves the turtle X steps forward. Negatives are prohibited.
2. BCK <steps>
Takes an Integer X and moves the turtle X steps backwards. Negatives are prohibited.
3. RGT <angle>
Rotates the turtle X degrees to the right. Valid values are
numbers from 8 to 359 Inclusive. Negatives are prohibited. 4. LFT <angle>
Rotates the turtle X degrees to the left. Valid values are
numbers from 8 to 359 Inclusive. Negatives are prohibited. 5. HOM .
Moves the turtle to the center of the screen.
6. CLS
Clears the screen. The turtle remains in the same position.

NOTE: We recommend using the goto-xy (LOCATE in BASIC) command to Implement this program.

Sample Input/Output

Input:

```
CLS
HOM
FWD 5
RGT 90
FWD 5
RGT '90
FWD 5
RGT 90
FWD 5
RGT 90
```

Output:

```
T · · · · ·
```

ICOM Challenge 1994
Programming Contest
March 5, 1994

Problem 3, Pascal to C Mini-For-Converter

Source File Name: AD3.XXX
Input File name: AD3.pas
Output File Name: AD3.c

Define: The for statement in the Pascal language has two formats:

For control-variable :- initial-value to final-value do program-statement;

```
ex.     for x:=1 to 5 do
            n:=n+x;
```

For control-variable :- initial-value down to final-value do program-statement;

```
ex.     for x:=5 down to 1 do
            n:= n + x;
```

The for statement in the C language has the following format:

```
For (initial-expression;loop-condition ; loop-expression)
    Program statement ;
```

```
ex.     for (x=1; x<=5; x++)
            n= n + x;

        for (x=5;x>=1; x - -)

            n= n + x;
```

The assignment statement in Pascal has the form

```
variable := expression ;
```

where expression may be a constant, another variable, or a formula to be evaluated.

. ~?)} ".

In C, the assignment statement has the following form:

variable - expression;

The begin and end statements in Pascal translate to open ({) and close (}) brackets in C. The end in Pascal has a semicolon (;), but the close bracket in C does not.

Problem: Write a program to translate valid for statements in Pascal to the C language. Your program will read a Pascal for statement from the file AD3.pas, and will write the equivalent C statement into the file AD3.c. Your translator program must consider the case of several assignment statements combined into a single for statement. You do not have to worry about nested for statements. You do not need to worry about arrays either.

Sample Input/Output:

Input:

```
for x:=I to 5 do begin
    n:=3;
    J:=n+x;
end;
```

Output:

```
for (x= I; x<=5; x++)
{
        n=3;
        J=n+x;
}
```

Problem 4. **Huffman Coding**

Source File Name: A D 4. X X X
Input File Name: A D 4. IN
Output File Name: A D 4. OUT

Define:

Huffman codes are used to compress data. One characteristics of Huffman codes is that the code words vary in the number of bits they contain. Data can be compresses efficiently by assigning the most frequently – ocurring data symbols to the shorter code words, and assigning the less frequently – ocurring data symbols to the longer code words.

Problem:

Write a program to read characters from the A D 4. IN , count the number of occurrences of each character, then assign Huffman code words to the characters as follows:

Huffman Code Word	Character
1	Most frequent
01	2nd most frequent
001	3rd most frequent
0001	4th most frequent
00001	5th most frequent
00000	6th most frequent

If the number of characters is the same for two different characters, assign the Huffman code words by the ASC l l numerical value of the character, considering the higher ASC l l value as "least frequent". You may assume that each character of the input file may be 1 of at most 6 different characters.

Then output the compressed version of the input file (l. e. replace each character by its Huffman code word) to the file AD4. OUT. For purposes of this problem, your output should be in the form of ASC l l "1" and "0" characters. (We realize that this means that you are not actually compressing the file, but it takes the problem easier to test and to check.)

Sample Input/Output:

Input:

ABCDEFDDDCDEDCCCEEBBF

Output:

000000001011001000011110110011010101001001000100010000 1

Input:

! ? # @ # ? # !! # ? # # ? ! ? @ # # ? !

Output:

00101100011011001001101110100101000111010 01

Problem 5. **Large Numbers**

Source File Name: AD5.XXX
Input File Name: AD5.1N
Output File Name: AD5.0UT

Problem: Write a program to multiply two large numbers of length up to 38 digits.

One approach is to treat each number as a list, each of whose elements is a block of digits of the number. Then you can multiple the integers (lists) element by element. Of course, you may use a different approach. Your program will read two numbers from the file AD5.1N. The numbers, represented by strings, will be in two separate lines. The output shall be written to file AD5.0UT.

Input:

Two Integers of length up to 39 digits represented by a string of ASCII characters to be entered Interactively bythe user.

Example:

To multiply 3452 * 2534, one could divide 3452 into 34 and 52, and 2534 into 25 and 34. Then,

52 * 34 = 1768, 34 * 34 = 1156,
25 * 52 = 1300, 25 * 34 = 850.

```
      1768       1300
  -----------  ------------
   117368     86300
```

And finally,

```
    1-17368
   +86300
 ----------------
    8747368
```

Note that for this last addition, the numbers may also be very large. Therefore, you will also need to separate each number into

```
+
115
6

+85
0
```

blocks of digits of the number, and then add the Integers (lists) block by block, carrying from one block to the next when necessary.
Note also that the resulting product can have more than 38 digits. You may assume that it will not have more than 58 digits.

Sample Input:

3452
2535

Sample Output:

The product is 8747368.

University of Puerto Rico
Mayaguez Campus

ICOM Challenge '94

Intermediate Division

Sponsored by

AEIC and Lucent Technologies

INTERMEDIA DIVISION

Problem1. STACK MANIPULATION

Source File Nam ID1.XXX

Input File Name: ID1.DAT

Output File Name: ID1.OUT

Write a program that converts an expression in infix notation to postfix and prefix notation. Your program should write as output the original expression in infix notation, the expression in postfix and in prefix notation. Each expression should be in a separate line, and properly labeled.

Sample Input:

(2+4/9)*3

Sample Output:

infix notation: (2+4/9)*3
postfix notation: 24 9+3*
prefix notation: +/24 9*3

Problem 2. **EASY CALENDAR**

Source File Name: **ID2.XXX**
Input File Name: **ID2.DAT**
Output File Name: **ID2.OUT**

Write a program that takes as input a date with the format:

mmmddyyyy

(yyyy can be any year before or after 1993)

and returns the day of the week corresponding to that date (i.e., Monday, Tuesday, etc.). Your program should also test for any invalid input, e.g. feb 29 1993 and report an error message in that case.

Remarks:
A given year is defined as a "leap" year if the year is divisible by 400, or if it is divisible by 4 but not by 100. For example, the year 2000 is a leap year; 1900 is not a leap year.

Sample Input:

feb 02 1993

Sample Output:

feb 02 1993 is a Tuesday

INTERMEDIATE DIVISION

EIGHT QUEENS WITH A TWIST

Problem 3:

Eight queens can be arranged on a chess board so that no Queen is under attack from any of the others; in other words, so that no row or column or diagonal contains more than one queen. Write a program that will place the position of 8 queens on a chess board and ensures that no Queens are under attack. Your program should first draw the chess board displaying the **X Y** matrix with Q's representing the Queen position. The program must also search and display all the solutions for this problem. The answers should show the correct solution.

Note: No hard-wired solution are allowed!

Sample Output:

	1	2	3	4	5	6	7	8
1	Q							
2			Q					
3								Q
4							Q	
5				Q				
6		Q						
7					Q			
8	Q							

PRESS ENTER FOR NEXT SOLUTION>

UNIVERSIDAD DE PUERTO RICO

RECINTO DE RIO PIEDRAS

Universidad de Puerto Rico
Recinto de Río Piedras

COMPETENCIAS DE PROGRAMACIÓN 1991

Source File Name: PRIN1.xxx
Input File Name: xxx
Output File Name: xxx

Direcciones generales:

Todos los problemas de esta categoría serán interactivos. Esto es, no leerán data de ningún archivo. Debes entregar el disco con tu programa tan pronto como lo hayas terminado. El equipo con mejor tiempo total (para resolver todos los problemas) será el ganador de entre los equipos que completen el mismo número de problemas satisfactoriamente.

PROBLEMA #1

ARCHIVO DE CODIGO: PRIN1.xxx

La serie K-generalizada de números Fibonacci esta definida como:

$$f_0 = f_1 = \ldots = f_{(k-1)} = 0 \text{ y } f_k = 1$$

$$f_{(n+k+1)} = f_{(n+k-1)} + \ldots + f_n \text{ para toda } n >= 0$$

Debes escribir un programa que lea pares de números enteros **n** y **k** y genere los primeros **n** número de la serie Fibonacci k-generalizada correspondiente.

INTERACCION:

Entre el parámetro k para la serie fibonacci k-generalizada
(0 para terminar): 3

Por favor entre la cantidad de elementos de dicha serie: 7

Los primeros 7 elementos de la serie fibonacci k-generalizada son:

Posición	**Número**
0	0
1	0
2	0
3	1
4	1
5	2
6	3

Entre el parámetro k para la serie fibonacci k-generalizada
(0 para terminar): 0

Gracias

PROBLEMA #2

ARCHIVO DE CODIGO: PRIN2.xxx

Escribe un programa que lea números enteros, **N**, y que calcule el factorial de **N** exactamente para cualquier entero N < 100. El factorial de un número está dado por

$$N! = 1 \times 2 \times 3 \times \ldots \times N$$

INTERACCION:

```
Favor de entrar un número entre 1 y 100 (0 para terminar): 5
        El factorial de 5 es 120
Favor de entrar un número entre 1 y 100 (0 para terminar): 10
        El factorial de 10 es 3628800
Favor de entrar un número entre 1 y 100 (0 para terminar): 0
        Gracias!
```

Source File Name: PRIN3.xxx

La función de Ackerman está definida como:

a (m , n) =	n + 1	cuando m = 0
a (m , n) =	a(m-1, 1)	cuando m <> 0 y n = 0
a (m , n) =	a(m-1, a(m, n-1))	cuando m <> 0 y n <> 0

Debes escribir una función que, dado dos números n y m, calcule a(m,n). Debes tabular los valores de a(m,n) para todas las m tal que $1 \leq m \leq 4$ y todas las n tal que $1 \leq n \leq 10$. EstePrgrama no tiene input.

Output

Función de Ackerman

n

m

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	**xx**	xx	xx	xx	xx	xx	xx	xx	Xx	xx
2	**xx**	xx	xx	xx	xx	xx	xx	xx	Xx	xx
3	**xx**	xx	xx	xx	xx	xx	xx	xx	Xx	xx
4	xx	xx	xx	xx	xx	xx	xx	xx	Xx	xx

PROBLEMA #4

ARCHIVO DE CODIGO: PRIN4.xxx

Debes escribir un programa que lea un número N entero impar positivo e imprima un cuadrado mágico NxN. Un cuadrado mágico NxN contiene los números enteros del 1 al N^2 de forma tal que las sumas de los elementos de cada fila, columna, y diagonal principal son iguales. Un algoritmo para construir un cuadrado mágico sería:

El algoritmo asume que los índices del cuadrado son del 1 al N en ambas dimenciones.

Paso I

Calcule F = Parte entera de (N+1) / 2
Calcule C = N

Paso II

Coloque 1 en la fila F columna C del cuadrado

PasoIII

En cada interación a continuación, coloque el siguiente número en la secuencia 2->N2 en la fila F columna C del cuadrado mágico
Incremente en cada interación ambas F y C en 1 (módulo N)= por N-1 veces. Coloque el próximo número de la secuencia.
En la siguiente interación reduzca F en 1 (módulo N) y no cambie C. coloque el próximo número en la fila F columna C.
Repita los pasos b y c hasta que haya colocado todos los números hata el N2 en el cuadrado mágico

=En este problema "módulo N" quiere decir que si x más (menos) 1 es igual a N +1 (0) entonces x pasa a ser 1 (N).

Ejemplo, si N = 3 y x = 3 entonces x + 1 módulo N es 1.
Ejemplo, si N = 3 y x = 1 entonces x – 1 módulo N es N.

INTERACCION:

Entre la dimención N del cuadrado mágico (0 para terminar): 3

Cuadrado mágico para N = 3

7	3	5
6	8	1
2	4	9

Entre la dimención N del cuadrado mágico (0 para terminar): 0
Gracias!

PROBLEMA #5

ARCHIVO DE CODIGO: PRIN5.xxx ,'

Un simple pero efectivo sistema de codificación depende de un arreglo de 25 letras del abecedario (inglés) en una matriz 5 x 5 dependiendo de una clave. Dicha clavedebe constar de 5 letras todas las cuales tienen que ser distintas. Se elige una letra de las 26 del abecedario para set transmitida intacta. Si la clave fuera BREAK y la letra a ser transmitida ihtacta fuera J, entonces la matriz cuadrada sería:

B	R	E	A	K
C	D	F	G	H
I	L	M	N	O
P	Q	S	T	U
V	W	X	Y	Z

Note que la letras son colocadas en secuencia a partir de la segunda fila, ig'norando aquellas letras que, o son parte de la clave, o son la letra a ser transmitida intacta.

El mensaje a ser codificado es descompuesto en segmentos de dos letras. Si el número de letras es impar, la útima letra es transmitida intacta. Carla letra de un par es codificada remplazandola con la letra que se encuentra en su misma fila de la matriz cuadrada, pero en la columna de la otra letra de la pareja. Si una pareja contiene la letra a ser transmitida intacta, ambas letras serán transmitidas intactas.

Escriba un programa que lea inicialmente una clave de 5 letras y una letra a ser transmitida intacta. Luego de verificar que la clave es legal (5 letras distintas) el programa debe leer mensajes de largo variable e imprimirlos codificados ignorando los espacios y signos de puntuación. Puede asumir que los mensajes son de no más de 80 letras.

INTERACCION:

Entre el código de 5 letras: BREAK
Entre la letra a ser transmitida intacta: J

Entre el mensaje a ser codificado (ENTER para terminar):
COMPUTER SCIENCE
El mensaje codificado es:
HiISTUREPFMBIGE

Entre el mensaje a ser codificado (ENTER para terminar):

[ENTER]
Gracias!

Universidad de Puerto Rico
Recinto de Río Piedras

COMPETENCIAS DE PROGRAMACIÓN 1991

Experto

EXPERT DIVISION

LONG, LONG DIVISION

Problem: 1

You have been assigned to a team of software-hardware engineers working on super computers development. Your task is to write software for implementing multiple digit division, which is to divide any integer of 40 or fewer digits by any positive divisor less than 100.

Data in the input files comes in pairs, with the first line containing the dividend and the second line containing the divisor. Your program is to accept only correct dividends and divisors. Thus, if either the dividend or its divisor contains any non-digit, i.e., a character not in [0..9], or the divisor is greater than 99 you are to print an error message, as shown in the sample out shown below.

If you read in two valid values, you are to compute the quotient and remainder and output the results as shown on the sample output shown below. You are to use normal end-of-file methods to terminate your reads from the input data file. Always skip a line, as shown below in the example data, between the dividend/divisor pairs.

Sample Data:

```
9
3
53
7
```

Sample output:

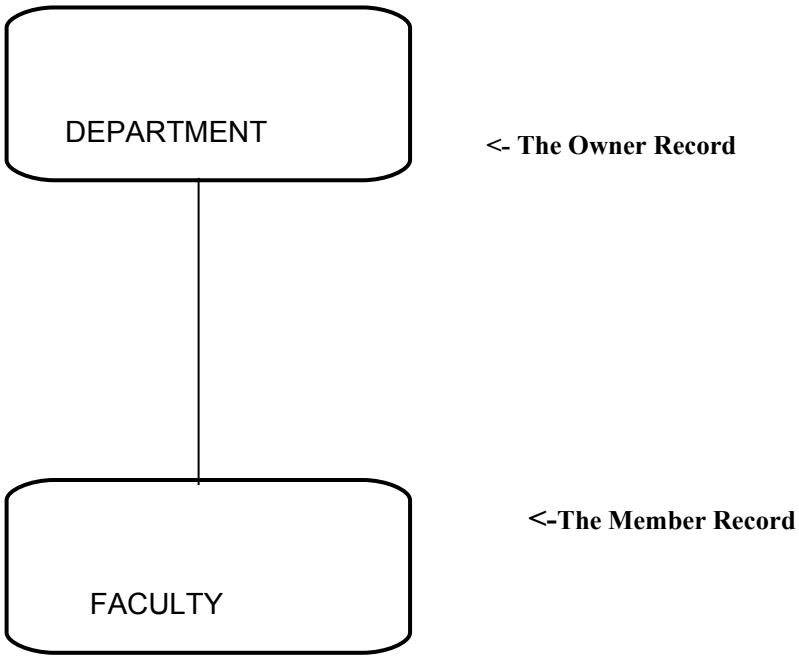
```
ENTER FIRST NUMBER> 9
ENTER SECOND NUMBER> 3
    Dividend is 9
    Divisor is 3
    Quotient is 3
    Remainder is 0

ENTER FIRST NUMBER> 53
ENTER SECOND NUMBER> 7
    Dividend is 53
    Divisor is 7
    Quotient is 7
    Remainder is 4

ENTER FIRST NUMBER> 0
<EXIT>
```

THE DATABASE PROBLEM

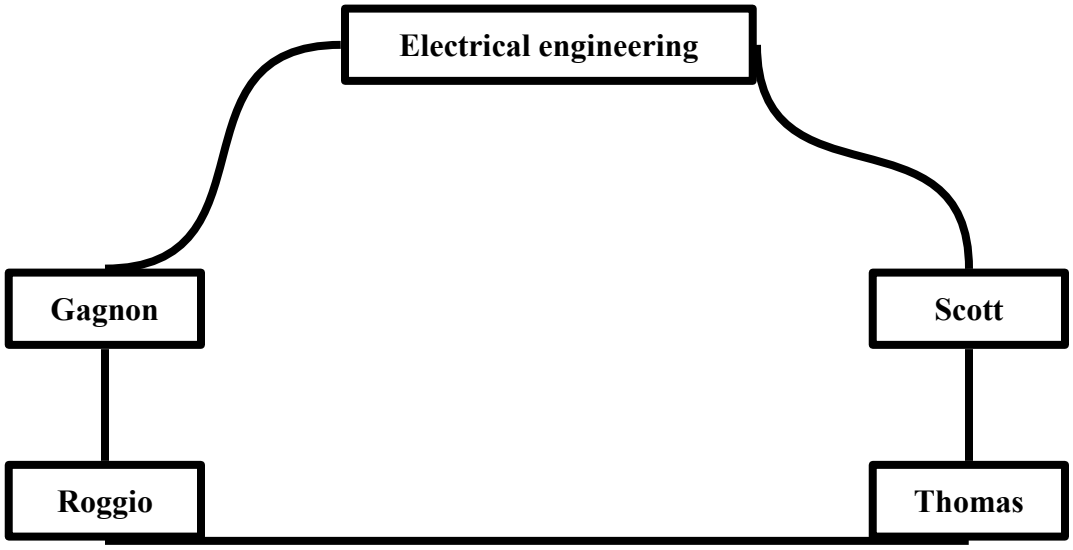
Source File Name: ED2.XXX
Input File Name: ED2.DAT
Output File Name: ED2.OUT

Define:

There is a standard database structure, called a set, which represents a two level hierarchy between two different record types – one called the owner and the other called the member. This construct is used to represent a 1 to many (i.e., 1:N) relationship between record types. The owner record is the 1 in the 1 :N relationship and the member is the N in the 1:N relationship. An example of this relationship (called a set type) is shown in the figure below. The figure below indicates a 1 :N relationship between the DEPARTMENT record and the FACULTY record. It co-notes that there can be N Faculty members associated with each department. Be aware that the N can be zero!



An occurrence of this set could be as follows:



Be aware that this is only one occurrence of the set type. There could be another occurrence for the Math Department, another occurrence for the Accounting Department, etc. Also note that the names in the DEPARTMENT record (Electrical Engineering) and the FACULTY records (SCOTT, THOMAS, ROGGIO, GAGNON) are used to indicate the record occurrences. There could be additional information (data items) concerning the DEPARTMENT (etc., Chairman´s Name, Department, Location, Budget, etc.) and the FACULTY (ex., Age, Salary, Address, etc.) stored in the record.

Problem:

You are to write a program to implement this set structure. Your program should accept unput records which contain various set commands. These commands are to produce various actions. These actions are summarized below, as well as the formats of each ot the commands described (format statement is written in bold). All input records can be coded in free field format. The only delimiters are blanks.

ADD DEPARTMENT – Add a new departmental (owner) record occurrence to the database.

ADD DEPARTMENT department – name department – budget

2. ADD FACULTY - Add a new faculty (member) record occurrence to the database. Each new faculty record must be associated with a departmental record

ADD FACULTY faculty – member – last-name salary deaprrment-name

3. LIST DEPARTMENT – list the names of all of the faculty members associated with a department.

LIST DEPARTMENT department-name

4. LIST FACULTY- List the name and salary of the specific faculty member.

LIST FACULTY faculty – member-last-name department-name

5. STOP- Stop proccesing.

STOP

Assumptions:

1. You may assume that **budget** and **salary** data are less that 60,000.
2. You may assume that all input data is in uppercase.
3. The maximum length of the **department name** and the **faculty name** is 20 characters.
4. You may assume that there will never be a LIST DEPARTMENT or a LIST FACULTY command for a DEPARTMENT or a FACULTY member that does not exist.
5. There will never be a LIST FACULTY command for a faculty member who is listed in the wrong department.
6. There will never be an ADD FACULTY command which contains a Department which does exist.

Output Requirements:

1. All input records should be echoed.
2. The output from the LIST FACULTY command should look as follows:

   FACULTY: faculty-name
   SALARY: salary

3. The output from the LIST DEPARTMENT command should look as follows:

   DEPARTMENT: department-name

BUDGET: budget

4. If there is an illegal command display the following message and continue processing.

   ERROR IN COMMAND

5. If there is an error in either the ADD or LIST commands, display the appropiate error message.

   ADD Command error
   LIST Command error

6. Double space all output except where noted. (See output requirements #2 and #3).

Sample Data:

ADD DEPARTMENT ELECTRICAL –ENG 50000
ADD DEPARTMENT INDUSTRIAL – ENG 30000
ADD FACULTY ESPOSITO 25000 INDUSTRIAL –ENG
ADD FACULTY GAGNON 22000 ELECTRICAL –ENG
ADD FACULTY GAGNON ELECTRICAL –ENG
LIST FACULTY GAGNON ELECTRICAL- ENG
LIST DEPARTMENT ELECTRICAL-ENG
STOP

Sample Output:

ADD DEPARTMENT ELECTRICAL –ENG 50000
ADD DEPARTMENT INDUSTRIAL –ENG 30000
ADD FACULTY ESPOSITO 25000 INDUSTRIAL –ENG
ADD FACULTY GAGNON 22000 ELECTRICAL –ENG
FACULTY: GAGNON
SALARY: 22000
DEPARTMENT: ELECTRICAL-ENG
BUDGET: 50000

Problem 3: **GOLDBACH CONJECTURE**

Source File Name: ED3.XXX
Input File Name: ED3.DAT
Output File Name: ED3.OUT

Define:

The Goldbach conjecture states that any positive even number greater than 4 can be expressed as the sum of two prime numbers. This conjecture has never been completely proven, but it has been demonstrated by computer to be true for a wide range of even numbers.

Problem:

Given an even number greater than 4, find two prime numbers which sum to it. For purposes of this problem, 1 is not considered a prime number.

Input:

Input for this problem consists of a list of even numbers greater than 4, one per line. The numbers will be fewer than 10 digits in length.

Output:

Each line of the program output consists of exactly three entities: the original input number and the two primes which sum to that number.

Sample Data:

8
10

Sample Output:

Original	Prime	
8	3	5
10	5	5

Problem 4: **CALCULATOR**

Source File Name: ED4.XXX
Input File Name: ED4.DAT
Output File Name: ED4.OUT

Problem:

You are to write a program which can evaluate simple expressions. The expression can contain the following t~-q;:ms:

decimal number:	optional sign, the digits 0-9 and decimal point,
operators:	+, -, *, or/,
parenthesis:	(or) to specify precedence grouping.

You can assume that the expression should be evaluated right to left, except where parenthesis grouping is found. You can also assume that all input expressions are syntactically correct and that each token is separated by exactly one blank space. All expressions will be less than 50 characters in length.

When an input expression is read, you are to print that expression and it's value correct to 6 decimal places.

Sample Data:

1■ +■ 2
2■* ■ 1.3 ■ / ■ (■77 ■* ■88 ■)

Note that ■ is equivalent to a space. **Sample Output:**

1 + 2 = 3.000000
2 * 1.3 / (77 * 88) = .000384

Problem 1 – **Path Finder**

Source File Name: ED1.xxx
Input File Name: ED1.DAT
Output File Name: ED1.Out

Problem:

Build a list from a set of input pairs identifying a start and end point of link on a path. Starting at point A trace the path to point E. Output the path that has the least links.

For example:
For the following pairs

A,B
B,C
C,D
D,E

The path from A to E is ABCDE.

Sample Data:
A,L
A,B
L,D
D,C
C,E
B,F
Sample Output:

The path from A to E is ABFE

Problem 2 – **The Yuca Complier**

Source File Name: ED2.xxx

Input File Name: ED2.DAT

Output File Name: ED2.Out

Definition:
A string is any combination of blanks, letters, digits and special character. Word are groups of characters separated by blanks.

Problem:
Develop a YUCA language parser/compiler. The YUCA (Yuyo's Uncle Compiler /Assembler) language can handle up to 20 variables, all of which are global. Variables can be up to 6 characters long. Only strings (up to 40 characters) can be assigned to variable. A variable can store strings up to 40 characters.

A YUCA programmer can use:

1) an INVERT function, which will reverse the string.(one argument)
2) a PRINT function, which will print the value for variable.(one argument)
3) a BEGIN symbol, which is used to mark the beginning of a program.
4) a END symbol, which is used to mark the end of a program.

All functions are reserved symbols. The YUCA compiler should print an error if:

+

Any function name is used as a symbol.
Any unassigned symbol is used in a function.
Any undefined function is used.

Also syntax errors should be flagged, BUT NOT DESCRIBED!!!. No nested functions are allowed in the YUCA language. NO NEED TO CHECK FOR CASE SENSITIVITY.

The compiler can read up to 5 programs as data.

A sample YUCA program is:

```
BEGIN
A1 ="BATATA";
A2 ="PLATANO";
A3 = "FOO";
INVERT (A3)

PRINT (A3);
PRINT (A1);
END

BEGON
```

```
A1 ="FOO";
END
```

and its output will be:

Program #1:
OOF
BATATA

Program # 2
Syntax Error at line 1

Problem 3 – **Movie Listings Database**

Source File Name: ED3.xxx
Input File Name: ED3. DAT
Output File Name: ED3.OUT

Definition:
A string is any combination of blanks, letters, digits and special characters.

Problem:
Read a set of data (times and movies) into a database and retrieve all movies playing between certain times and retrieve all times for the movie "ROCKY 55" as the last query. The retrieve query has the format:

R<start-time><end-time>

<u>No data will follow any query.</u> (No need for error checking) The indicator for the beginning of queries will be an empty line.

The time will be the first 5 characters of the input line, followed by a comma (,) and a string of less than 15 characters containing the name of the movie.

<u>The program should handle incorrect input data.</u> Data checks should be made for illegal times.

For example:
In the following input:

7:30, ROCKY 55
9:00, T2
10:00, BLUE VELVET

R 7:30 9:00

The output should be:
Playing between 7:30 and 9:00 are ROCKY 55, T2
ROCKY 55 being played at:7:30.

Problem 4 – **Directory Search**

Source File Name: ED4.xxx
Input File Name: ED4.DAT
Output File Name: ED4.OUT

Definition:
Words are groups of characters separated by blanks.

Problem:
Write a program to read a :"mutilated" word representing a name and substitute the closest matching "corrected" names from a list of "correct" names. To determine the closest matching name, assume the first letter has to match, the word has to be within 2 characters of the selected " correct" name, and the selected "correct" name has the most character matches with the mutilated name (count only matching characters regardless of position).

Assume that names contain only capital letters. The input file will consist of a list of up to 8 "corrected" names, followed by up to 20 "incorrect" names. Both of these lists will be separated by an empty line.

For example, for an input of:

JENNIFER
TRACY
JEANINE
SANDRA
JOSEPH

JENIFER
JOSE
TRAPPER

The output should be:

Match for JENIFER is JENNIFER
Match for JOSE is JOSEPH
Match for TRAPPER is TRACY

Problem 5- **Puzzle**

Source File Name: ED5.xxx
Input File Name: ED5.DAT
Output File Name:ED5.OUT

Problem: You are to write a program which will scan a 10x10 character grid, and find the occurrence of a particular word in the grid. The word can be found horizontally, vertically, diagonally ON ANY DIRECTION.

For instance, given a grid: (4x4)

A	**B**	**D**	**X**
K	O	H	P
H	W	U	P
P	L	U	P

and the next input being:
BOWL
PULP
PULL

he program should print the location of the word, like:

BOWL is in positions, (2,1), (2,2), (2,3), (2,4).
PULP is in positions (1,4), (2,4), (3,4), (4,4)
The word PULL is not in the grind.

Problem 6- The “Pyramid” Sort

Source File Name: ID2.xxx
Input File Name: ID2.DAT
Output File Name: Id2.OUT

Definition:
A string is any combination of blanks, letter, digits and special characters. Words are groups of characters separated by blanks.

Problem:
Write a program that will read a list of integers (up to 30), and sort them placing the largest number in the middle, the second largest to the left of the list, third larges to the right of the list, and so on.

For example, if we sort:
5, 4, 8, 20, 12

we will have:
5, 12, 20, 8, 4

The program should work for both even and odd number of elements in the list. For even numbers, the largest number should be placed in the N/2 + 1 position, the following with the same pattern.

For example: 1, 2, 3, 4, 5, 6

Will be sorted as: 1, 3, 5, 6, 4, 2

UNIVERSIDAD INTERAMERICANA DE PUERTO RICO

RECINTO DE BAYAMÓN

7mas OLIMPIADAS DE PROGRAMACION
INTERBAY 2006

EXPERTOS

Problem: Matrix Matcher

Input: Standard Input
Output: Standard Output

Given an N * M matrix, your task is to find the number of occurrences of an X * Y pattern.

Input

The first line contains a single integer t ($t \leq 15$), the number of test cases. For each case, the first line contains two integers N and M ($N, M \leq 1000$). The next N lines contain M characters each. The next line contains two integers X and Y ($X, Y \leq 100$). The next X lines contain Y characters each.

Output

For each case, output a single integer in its own line, the number of occurrences.

Sample input

```
2
1  1
x
1  1
y
3  3
abc
bcd
cde
2  2
bc
cd
```

Output for Sample Input

```
0
2
```

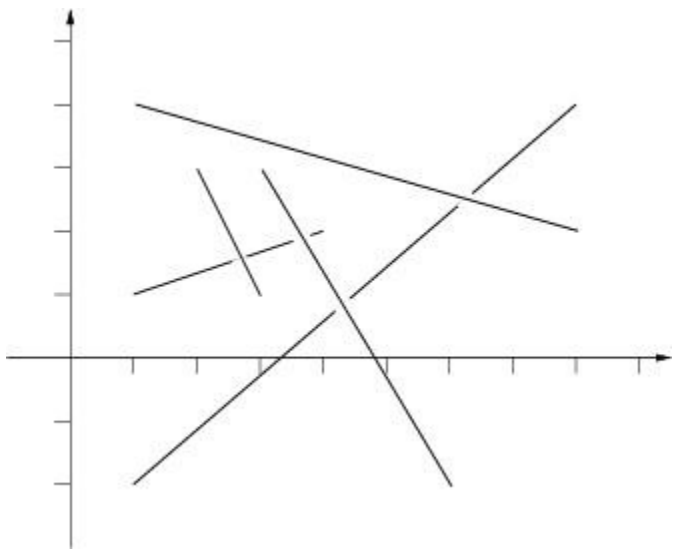
Problem : Pick-up sticks

Stan has *n* sticks of various lengths. He throws them one at time on the floor in a random way. After finishing throwing, Stan tries to find the top sticks that are these sticks such that there is no stick on top of them. Stan has noticed that the last thrown stick is always on top but he wants to know all the sticks that are on top. Stan sticks are very, very thin such that their thickness can be neglected.

Input consists of a number of cases. The data for each case start with $1 \le n \le 10000$, the number of sticks for this case. The following *n* lines contain four numbers each; these numbers are the planar coordinates of the endpoints of one stick. The sticks are listed in the order in which Stan has thrown them. You may assume that there are no more than 1000 top sticks. The input is ended by the case with $n = 0$. This case should not be processed.

For each input case, print one line of output listing the top sticks in the format given in the sample. The top sticks should be listed in order in which they were thrown.

The picture to the right below illustrates the first case from input.



Sample Input

```
5
1 1 4 2
2 3 3 1
1 -2.0 8 4
1 4 8 2
3 3 6 -2.0
3
0 0 1 1
1 0 2 1
2 0 3 1
0
```

Output for Sample Input

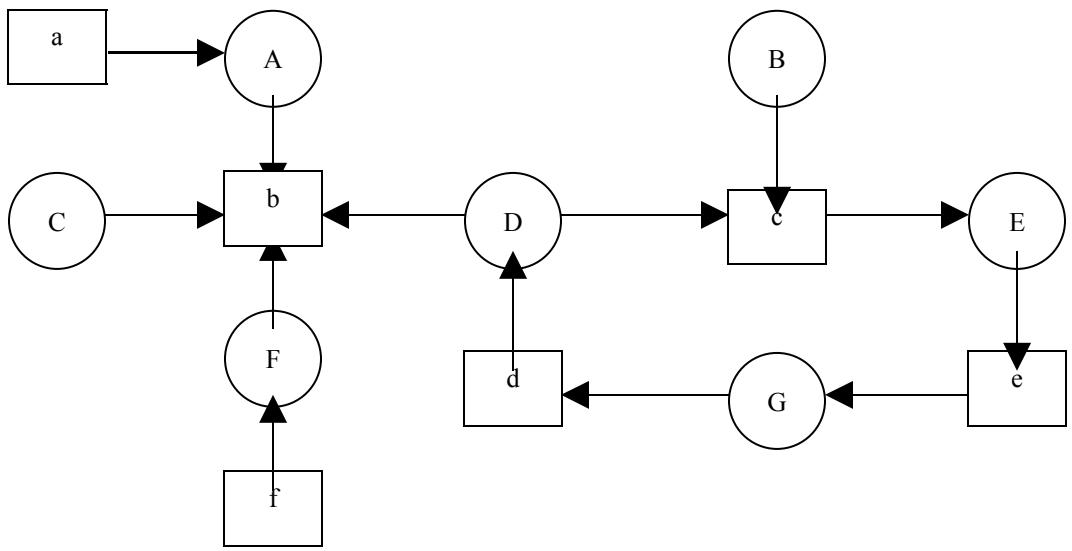
```
Top sticks: 2, 4, 5.
Top sticks: 1, 2, 3.
```

Deadlock Detection

In Operating Systems a special resource-allocation graph algorithm can be used to detect whether there is any deadlock in the system. A resource-allocation graph is a directed graph consisting of two different types of nodes $\boldsymbol{P} = \boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_n$, the set consisting of all resource types in the system.

A directed edge from process $\boldsymbol{P}_i$ to resource $\boldsymbol{R}_j$, is denoted by $\boldsymbol{P}_i \rightarrow \boldsymbol{R}_j$ and means that process $\boldsymbol{P}_i$ requested an instance resource type $\boldsymbol{R}_j$, and is currently waiting for that resource. A directed edge from resource type $\boldsymbol{R}_j$ to process $\boldsymbol{P}_i$, is denoted by $\boldsymbol{R}_j \rightarrow \boldsymbol{P}_i$ and means that an instance of resource type $\boldsymbol{R}_j$ has been allocated to process $\boldsymbol{P}_i$.

The following figure illustrates a resource-allocation graph where processes are denoted by circles and resources by squares. Notice that if there is a circular wait among the processes, then it implies that a deadlock has occurred.



Sequence: D-c-E-e-G-d-D

Given a resource allocation graph in which each resource type has exactly one instance, your job is to do determine whether there is a deadlock in the system. In case a deadlock exists, you must also show the sequence of processes and resources involved.

INPUT

The input begins with a single positive integer on a line by itself indicating the number of cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

We will assume that the processes are named by capital letters and resources by small letters, so we limit to 26 the number of processes and/or resources. Therefore, the first line of input consists of three numbers ***N***, ***M*** and ***E***, respectively, the number of processes, the number of resources and the number of edges. The edges are given in the following lines as pairs of letters linked by a `-' character. Edges are separated by spaces or newlines.

OUTPUT

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output must be `NO' if no deadlock is detected. In case e deadlock is detected, the output must be `YES' followed by the sequence or sequences of circular waits detected, one per line. If more than one sequence is found, they should all be output in an increasing of their length.

Sample Input

```
1

2 2 4
A-b B-a
a-A b-B
```

Sample Output

```
YES
A-b-B-a-A
```

Document Validator

Nowadays mark-up languages like `LaTeX`, `HTML`, `SGML`, and `XML` are widely used to define, in a completely textual mode, the structures of documents. Those languages are composed by tags, or annotations, that are used to mark blocks of text (their begin and end) in order:

- to specify the document structures- for instance, the *front material* like the *title*, the *authors*, the *date*, or a *chapter*, with its *sections* and *paragraphs*;
- to give them some special interpretation, or a particular formatting information – for instance, to say that the block designates a *country*, a *profession*, a *king*, or the block must be printed out in *italic*, or *boldface*.

Tags are just words like all the others that belong to the original plain text; so, it is necessary to use some lexical convention to distinguish those special words (annotations). Also, we need some lexical convention to identify the beginning and the end of the text block we want to annotate.

In the present context, our mark-up language follows an approach similar to the `SGML` based languages:

- special characters – square `[ ]` and curly `{ }` brackets – are used to identify the words that are tag names;
- the same tag name is used to setup block boundaries, and lexical detail defines the opening and closing tag. The convention is: `[opening-tag [,] closing-tag]` and `{opening-closing-tag}`.

Paired tags are employed to define the document structure, while single tags are used to give meaning or formatting information. Tag names (one letter followed by zero or more letters or digits) are free, i.e. not previously defined; this means that the user of that mark-up language can chose the names of the tags he will employ to annotate his document. However, to be a **valid annotation** it must be in conformity with the following rules:

- an *opening-tag* must always have a corresponding *closing-tag*;
- a *closing-up* must always have a corresponding *opening-tag*;
- paired tags may be nested to any level, however the last opening-tag must be closed before any enclosing tag, i.e. one external block can not be closed before an internal one;
- an *opening-closing-tag* can not have other tags inside; in that case, the corresponding block of text will appear inside the curly brackets.

Given a structured document, supposed to be annotated accordingly to the mark-up language above described, write a program that validates it, that is that verifies if the tags are properly

Input

The input begins with a single positive integer on a line by itself indicating the number of cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.
The input is a plain text file with tags, following the above stated conventions. Assume that lexical conventions specified for the three tag types (`[opening-tag [,] closing-tag]` and `{opening-closing-tag}`) are always complied.

Output

For each case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output consists of:

- just 1 line with the word `error' if one of the rules of above is not observed, making the input an invalid annotated document;

- 2 lists with the tag names found, without duplicates and in the order they appear in the document (from the beginning to the end), 1 word per line. The first list begins with the heading line `structural tags', while the second list will begin with the heading line `semantic tags. Blank lines between those are no allowed.

Sample input

```
1

[memo[
[de[ Comiss ao Cient fica do MIUP ]de]
[para[ Todos os Concurrentes ]para]
[data[{bold 2001.set.25}]data]
[mensagem[
[parag[Devem ter o m axiom cuidado na leitura dos enunciados.]parag]
[parag[Desejamos a todos {desejo Calma} e {desejo Boa Sorte}!]parag]
]mensagem]
]memo]
```

Simple Output

```
STRUCTURAL TAGS
memo
de
para
data
mensagem
parag
SEMANTIC TAGS
bold
desejo
```

7mas OLIMPIADAS DE PROGRAMACION
INTERBAY 2006

Intermedio

Problema #_____	CAMINO MAS CORTO
Autor:	Prof. José A. Rodríguez Ortega

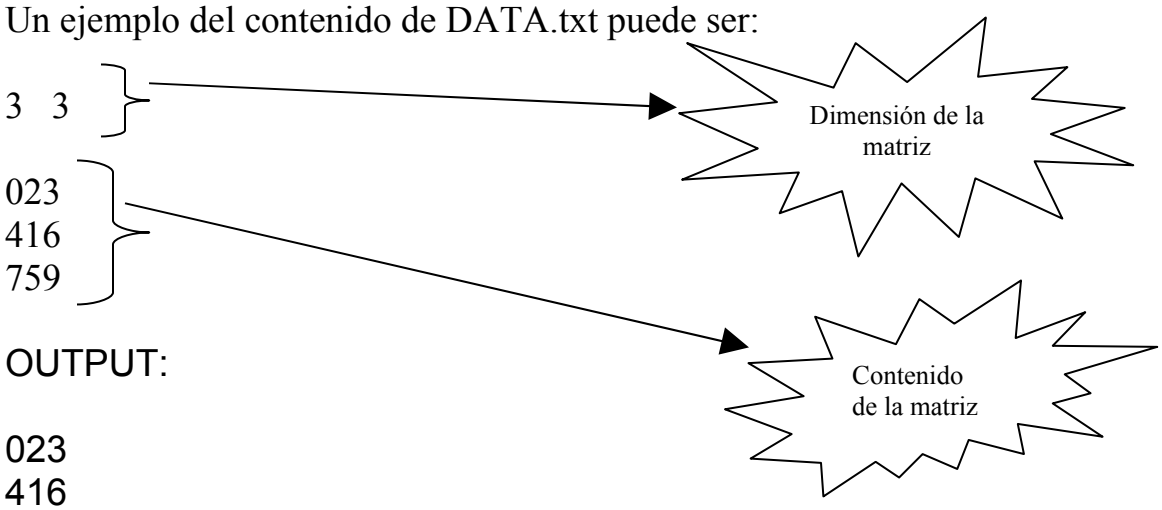
Desarrolle un programa en el leguaje de su preferencia, para que lea un archivo llamada DATA.txt, que contendrá una matriz mayor o igual a 2 X 2. Su programa deberá leer la matriz y determinar la ruta más corta para llegar desde la casilla (0,0) a la (m,n) es decir: desde el extremo superior izquierdo hasta el inferior derecho (en este caso desde el 1 hasta el 16).
Ej.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

En este ejemplo la ruta más corta es: 1,2,3,4,8,12,16

INPUT:

Un ejemplo del contenido de DATA.txt puede ser:



OUTPUT:

023
416
759

La ruta más corta entre 0 y 9 es: 0, 2, 1, 5, 9

Nota: Debe tomar en consideración que su programa debe reconocer cualquier matriz mayor o igual a 2 X 2. Ejemplos: 3 X 3, 4 X 4 etc.

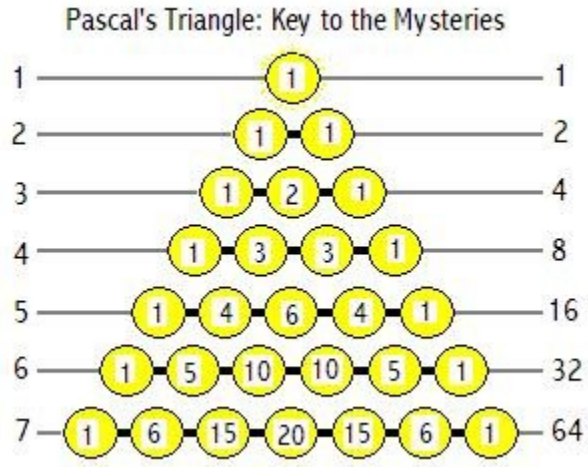
Problema #_____	DIAMANTE DE PASCAL
Autor:	Prof. José A. Rodríguez Ortega

Implemente un diamante basándose en el Triángulo de Pascal. El usuario entrará el nivel y su programa deberá generarlo. El diamante debe ser nivel 3 en adelante.

Ejemplo:

Entre el nivel de su diamante: 6

Ejemplo 2:
Entre el nivel de su diamante: 7



Nota: Su programa mostrará solo el dimante, no incluya los niveles.

Problema #_____	TIC TAC TOE 3D Interactivo
Autor:	Prof. José A. Rodríguez Ortega

Implemente un juego de Tic Tac Toe interactivo donde el usuario pueda jugar contra el código creado por usted. Debe tomar en consideración que se puede ganar horizontal, vertical y diagonalmente. Su programa deberá ser inteligente, es decir, la computadora debe programarse para ganar y jugar estratégicamente. No debe hacer meramente jugadas aleatorias.

Problema #_____	VEREDAS CON PROPÓSITO
Autor:	Prof. José A. Rodríguez Ortega

Desarrolle un programa en el lenguaje de su preferencia, para que lea un archivo llamado DATA.txt que contiene un valor entero cualquiera y una 3 X 3 con datos como los siguientes:

INPUT:

25 → número entero arbitrario

785
632
149 } → contenido de la matriz

Su programa deberá navegar todas las rutas posibles desde el extremo superior izquierdo (0,0) 7 en este caso, hasta el extremo inferior derecho (n, m) 9 en este caso. Su programa deberá mostrar todas las rutas posibles **sin repetir casillas**, la ruta de la sumatoria más cercana al número que se provee en el archivo (25 en este caso). Los caminos se pueden construir moviéndose hacia arriba, abajo, a la derecha y a la izquierda nunca en diagonal. No es necesario que siga el siguiente orden pero tienen que estar todas las veredas.

OUTPUT:

7 + 8 + 5 + 2 + 3 + 6 + 1 + 4 + 9 = 45
7 + 8 + 5 + 2 + 3 +4 +9 = 38
7 + 8 + 5 + 2 +9 = 31
7 + 8 +3 + 6 + 1 + 4 + 9 = 38
7 + 8 +3 + 2 +9 = 29
7 + 8 +3 + 4 +9 = 31
7 + 6 +3 + 8 + 5 + 2 +9 = 40
7 + 6 +3+ 2 +9 = 27
7 + 6 +3 + 4 +9 = 29

La ruta más cerca de [25] es: 7 + 6 +3+ 2 +9 = 27

Nota:

Los valores de la matriz pueden contener valores negativos. De existir rutas que produzcan un mismo resultado, el programador escogerá a conveniencia la que quiera mostrar.

Problem A.Ant on a Chessboard

Background

One day, an ant called Alice came to an M*M chessboard. She wanted to go around all the grids. So she began to walk along the chessboard according to this way: (you can assume that her speed is one grid per second)

At the first second, Slice was standing at (1, 1). Firstly she went up the grid, then a grid to the right, a grid downward, After that, she went a grid to the right, them two grids upward, and then two grids to the left in a word, the path was like a snake.

For example, her first 25 seconds went like this:
(the numbers in the grids stands for the time when she went into the grids)

25	24	23	22	21	5
10	11	12	13	20	4
9	8	7	14	19	3
2	3	6	15	18	2
1	4	5	16	17	1
1	2	3	4	5	

At the 8th second, she was at (2, 3), and at 20th second, she was at (5, 4).
Your task is to decide were she was at a given time.
(you can assume that M is large enough)

Input

Input file will contain several lines, and each line contains a number N(1<=2*10^9), which stands for the time. The file will be ended with a line that contains a number 0.

Output

For each input situation you should print a line with two numbers (x, y), the column and the row number, there must be only a space between them.

Sample Input

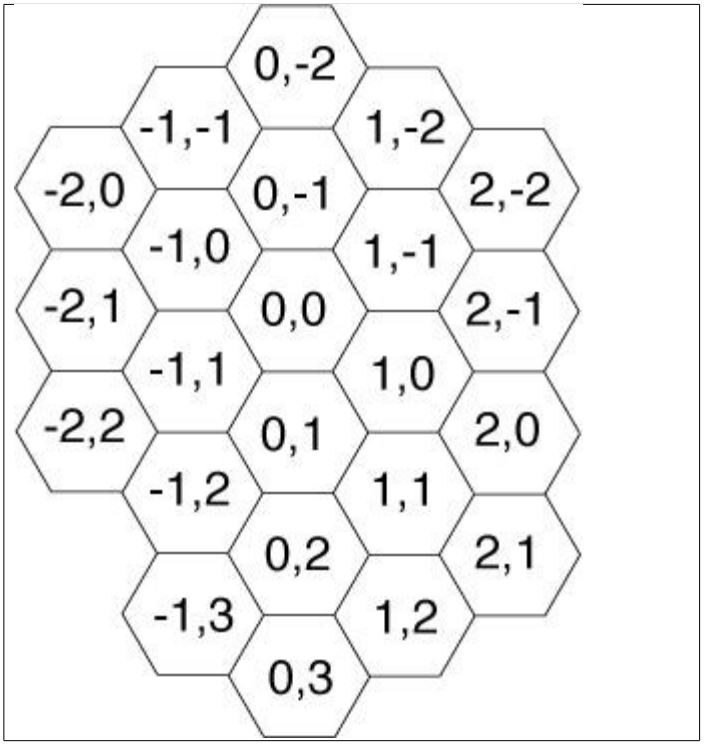
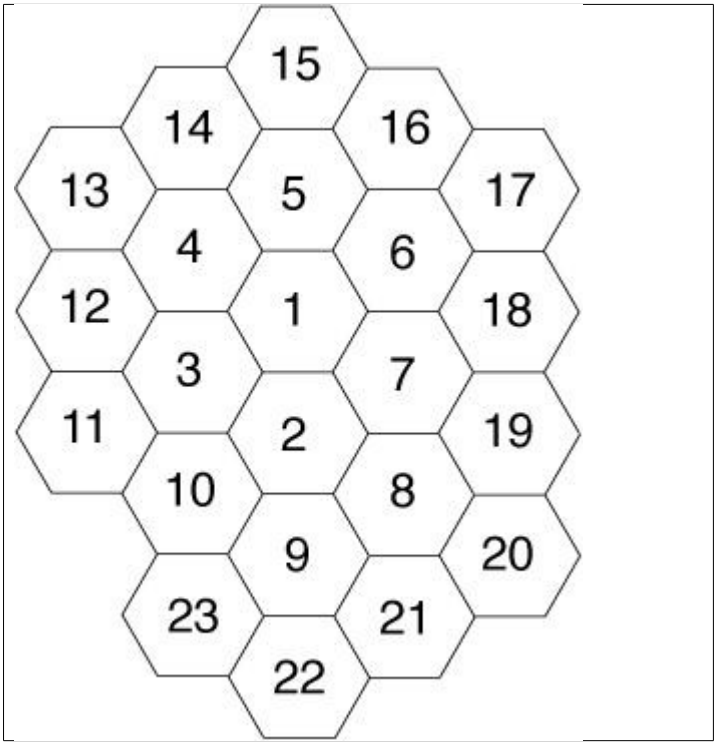
```
8
20
25
0
```

Sample Output

```
2 3
5 4
1 5
```

Problem A: Bee Maja

Maja is a bee. She lives in a bee hive with thousands of other bees. This bee hive consists of many hexagonal honey combs where the honey is stored in.
But bee Maja has problem. Willi told her where she can meet him, but because Willi is a male drone and Maja is a female worker they have different coordinate systems.

Maja's Coordinate System	Willi's Coordinate System
Maja who often flies directly to a special honey comb has laid an advanced two dimensional grid over the whole hive.	Willi who is more lazy and often walks around just numbered the cells clockwise starting from 1 in the middle of the hive.
	

Help Maja to convert Willi's system to hers. Write a program which for a given honey comb number gives the coordinates in Maja's system.

Input Specification

The input file contains one or more integers which represent Willi's numbers. Each number stands on its own in a separate line, directly followed by a newline. The honey comb numbers are less than 100 000.

Output Specification

You should output the corresponding Maj coordinates to willi's numbers, each coordinate pair on a separate line.

Sample Intput

```
1
2
3
4
5
```

Sample Output

```
0   0
0   1
-1   1
-1   0
0   -1
```

7mas OILIMPIADAS DE PROGRAMACION
INTERBAY 2006

PRINCIPIANTES

1. **Lowercase To Uppercase Convert**

Write a program that lets the user enter a string a character array. The program should then convert all the lowercase letters to uppercase or to Title Case. (If a character is already uppercase, or is not a letter, it should be left alone.) *Hint:* Consult the ASCCI chart in Appendix A. Notice that the lowercase letters are represented by the ASCII codes 97 thorough 122. If you subtract 32 from any lowercase character's ASCII code, it will yield the ASCII code of the uppercase equivalent.

Example:

This program is difficult → THIS PROGRAM IS DFFICULT → This Program Is Difficult

2. **Driver's License Exam**

The local Driver's License Office has asked you to write a program that grades the written portion of the driver's license exam. The exam has 20 multiple questions. Here are the correct answers:

1. B	6. A	11. B	16. C
2. D	7. B	12. C	17. C
3. A	8. A	13. D	18. B
4. A	9. C	14. A	19. D
5. C	10. D	15. D	20. A

Your program should store the correct answers shown above in an array. It should ask the user to enter the student's answers for each of the 20 questions, which should be stored in another array. After the student's answers have been entered, the program should display a message indicating whether a student passed or failed the exam. (A student must correctly answer 15 of the 20 questions to pass the exam.) It should then display the total number of correctly answered questions, and a list showing the questions numbers of the incorrectly answered questions.

*Input validation: Only accept the letters **A, B, C** or **D** as answers*.

3. **Lottery Application**

Write a program that simulates a lottery. The program should have an array of five integers named lottery, and should generate a random number in the range of 0 through 9 for each element in the array. The user should enter five digits which should be stored in an integer array named user. The program is to compare the corresponding elements in the two arrays and keep account of the digits that match. For example, the following shows the lottery array and the user array with sample numbers stored in each. There are two matching digits (elements 2 and 4).

Lottery array:

7	4	9	1	3

User array:

4	2	9	7	3

The program should display the random numbers stored in the lottery array and the number of digits matching digits. If all the digits match, display a message proclaiming the user as a grand prize winner.

4. **(Cash Register Application)** Use the numeric keypad from the **Security Panel** application to build a **Cash Register** application. In addition to numbers, the cash register should include a decimal point Button. Apart from this numeric operation, there should be **Enter, Delete, Clear** and **Total** Buttons. Sales tax should be calculated on the amount purchased. Use a Select Case statement to compute sales tax. Add the tax amount to the subtotal to calculate the total. Display the tax and total for the user. Use the following sales-tax percentages, which are based on the amount of money spent:

Amount under $100 = 5% (.05) sales tax
Amount between $100 and $500 = 7.5% (.075) sales tax
Amount above $500 = 10% (.10) sales tax

a) **Define event handlers for the numeric Buttons and decimal point in the keypad.** Create event handlers for each of these Button's Click events. Have each event handler concatenate the proper value to the TextBox at the top of the Form.
b) **Define an event handler for the Enter Button's Click event.** Create an event handler for this Button's Click event. Have this event handler add the current amount to the subtotal and display the new subtotal.
c) **Define an event handler for the Total Button's Click event.** Create an event handler for this Button's Click event. Have this event handler use the subtotal to compute the tax amount.
d) **Define an event handler for the Clear Button's Click event.** Create an event handler for this Button's Click event. Have this event handler clear the user input and display the value $0.00 for the subtotal, sales tax and total.
e) **Define an event handler for the Delete Button's Click event.** Create an event handler for this Button's Click event. Have this event handler clear only the data in the TextBox.

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2004

Expertos

Universidad Interamericana de Puerto Rico
Quintas Olimpiadas de Programación
INTERBAY 2004

CATEGORÍA DE EXPERTOS

Instrucciones generales:

Todos los problemas de esta categoría serán interactivos y/o con archivos externos. Se utilizará Visual Studio. Net para resolver los mismos. Deben entregarse en disco tan pronto como lo hayas terminado.

PROBLEMA 1

Decimal-Binary

A simple method for converting base 10 decimals to base 2 is to repeatedly multiply by 2 and take the integer result. For example, the conversion of .140625 base to .001001 base 2 is shown by the following chart:

.140625 0.28125 =>0
.28125 0.5625 =>0
.56251.125 =>1
.125 0.25 =>0
.25 0.5 =>0
.5 1 =>1

This process terminates when the decimal portion of the product is 0.

There will be 5 numbers input, each less than 1.0. Convert each input to binary and print the first 10 digits of the base 2 number, or fewer if the process terminates before that. Do not print any digits to the left of the "decimal point."

Sample Input:

Line 1: .140625
Line 2: .111

Sample Output:

Output 1: .001001
Output 2: .0001110001

Programming Problem #1

Test Data Input:

#1: .105
#2: .45
#3: .25
#4: .125
#5: .001

Test Data Output:

#1: .0001101011
#2: .0111001100
#3: .01
#4: .001
#5: .0000000001

PROBLEMA 2

Time Card

Jenny just started work as a programmer for Justine's Java Workshop. She is paid $10 an hour, with a few exceptions. She earns an extra $1.50 an hour for any part of a day where she works more than 8 hours, and an extra $2.50 an hour for hours beyond 40 in any one week. Also, she earns a 25% bonus for working on Saturday and Sunday are computed based on the hours worked those days; they are not used to calculate any bonus for working more than 40 hours in a week.

You'll be given the number of hours Jenny worked each day in a week (Sunday, Monday, …, Saturday), and you need to compute her salary for the week. The input will be positive integers, less than or equal to 24. The output must be formatted with a dollar sign and rounded to the nearest penny. For example, "$2" and "$2.136666" are wrong answers; the correct versions are " $2.00" and "$2.14", respectively. There may not be any embedded spaces in your answers. There will be 5 sets of data.

Sample Input:

Line 1:0, 8, 8, 8, 10, 6, 0
Line 2:4, 0, 0, 0, 0, 6, 0

Sample Output:

Output 1: $403.00
Output 2: $120.00

PROBLEMA 3

Multiplicación Rusa

En el siglo pasado en Rusia, los campesinos se servían de un ingenioso método de multiplicación para el que sólo necesitaban saber encontrar el doble y la mitad de un número, así como la elemental operación de sumar. Con este sistema había que formar dos columnas encabezadas por las cifras que se iban a multiplicar.

El método opera en la siguiente forma:

Ejemplo: multiplíquese 97 por 39

Se saca sucesivamente la mitad a los números situados en la columna de la izquierda(sin tomar en cuenta los residuos), y el doble a los de la columna derecha. Se continúa esta operación hasta que la columna izquierda se reduzca a la unidad.

97	X	39
48	X	78
24	X	156
12	X	312
6	X	624
3	X	1248
1	X	2496

Se tachan todas las cifras pares que figuren en la columna izquierda y por consiguiente las cifras correspondientes en la columna derecha.

9	X	39
7		
4	X	78
8		
2	X	156
4		
1	X	312
2		
6	X	624
3	X	1248
1	X	2496

Finalmente se suman las cifras que quedan en la columna derecha. El total de la suma es el resultado. 39+1248+2496=3783

PROBLEMA 4

Vertical Histogram

Escribe un programa que acepte dígitos (0-9) como Input. De como resultado el histograma vertical representativo de cada dígito. Calcula también la suma de todos los dígitos, el promedio, y da como resultado la lista de los números por debajo del promedio sin repetir y la lista de los números por debajo del promedio sin repetir y la lista de los números por encima del promedio sin repetir.

Verifica tu programa con este set de 13 dígitos.
1,7,2,9,6,7,1,3,7,5,7,9,0

Ejemplo de Input:

```
Enter a Number :12
Enter 12 digits:
1,7,2,9,6,7,1,3,7,5,7,9
```

Ejemplo de Output:

```
         *
         *
 *       *    *
 ***  ***     *

0123456789
```

La suma es : 64 El promedio es : 5.3
Los números por debajo son: 1,2,3
Los números por encima: 6,7,9

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890

1
121
12321
1234321
12354321
12345654321
1234567654321
123456787654321
12345678987654321
123456787654321
1234567654321
12345654321
123454321
1234321
12321
121
1

==== End of output data
```

Problema 5

Virus Detection

When examining a file, a certain virus scanner looks for, among other things, occurences of word “virus” in the file. For example, the line.

ab12$virus23

---

appearing in some file would be marked as suspect, as it contains an occurrence of the string “virus”. It is safer, however, to also look for what are called “scattered occurrences” of the string “virus”. For example, the line.

alirusv2vriirc$u5#us9

- - - -

contains a “scattered occurrence” of the string “virus”, which is indicated by the underlined characters. Note that even in a “scattered occurrence” of the word the order of the letters remains the same as in the correctly spelled word “virus”.

Your task in to write a program that reads each line in a textfile called TEST.DAT, and outputs the line number of every line of text which contains either a direct occurrence of the string "virus", or a "scattered occurrence" of the string "virus", as described above. The first line of the file contains just the number of lines of text to follow. The lines of text in the file from line 2 of the file onward have "line numbers" starting at 1.

```
==== Sample input data from VIRUS1.DAT

4
dsffdvi5wqdhdr
vvvvirus23e4
@#1vviwdscr32uu2ss
%%gfsd
====End of input data/Star of corresponding output data
====================

2
3
====End of output data

=



====Sample input data from VIRUS2.DAT
=============================================

6
Itisveryinvigoratingtorunaroundtheskyscraper.
This line does not contain the v*i*r*u*s string.  Or does it?
)(^*%&V#@$@i*)*)&R879078u*)*)*S
Nowisthetimeforallhackersandvvvviiirrruuussswriterstoceaseanddesist!
Fajlcvjioryweoprhavkahyiesdjvoefhbg
Piroutrgbsfoguofgnktnhnskhkhkhkhkhbfskklhfeikbmadfgiyewt
====End of input data/Start of corresponding output data
=====================
1
2
4
====End of output data
```

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2004

Principiantes

Universidad Interamericana de Puerto Rico

Quintas Olimpiadas de Programación
INTERBAY 2004

CATEGORÍAS DE PRINCIPIANTES

Instrucciones generales:

Todos problemas de esta categoría serán interactivos y/o con archivos externos. Se utilizará Visual Studio .Net para resolver los mismos. Deben entregarse en disco tan pronto como lo hayas terminado.

PROBLEMA 1

(Cash Register Application)

Use the numeric keypad from the Security Panel application to build a Cash Register application. In addiction to numbers, the cash register should include a decimal point Button. Apart from this numeric operation, there should be Enter, Delete, Clear and Total Buttons. Sales tax should be calculated on the amount purchased. Use a Select Case statement to compute sales tax. Add the tax amount to the subtotal to calculate the total. Display the tax and total for the user. Use the following sale-tax percentages, which are based on the amount of money spent:

Amount under $100= 5% (.05) sales tax
Amount between $100 and $500= 7.5% (.075) sales tax
Amount above $500 = 10% (.10) sales tax

Define event handlers for the numeric Buttons and decimal point in the keypad. Create event handlers for each of these Button's Click events. Have each events handler concatenate the proper value to the TextBox at the top of the Form.

Define an event handler for the Enter Button's Click event Create an event handler for this Button's Click event. Have this event handler add the current amount to the subtotal and display the new subtotal.

Define an event handler for the Total Button's Click event Create an event handler for this Button's Click event. Have this event handler use the subtotal to compute the tax amount.

Define an event handler for the Clear Button's Click event Create an event handler for this Button's Click event. Have this event handler cleat the user input and display and display the value $0.00 for the subtotal, sales tax and total.

Define an event handler for the Delete Button's Click event Create an event handler for this Button's Click event. Have this event handler clear only the data in the TextBox.

PROBLEMA 2

(Present Value Calculator Application)

A bank wants to show its customers how much they would need to invest to achieve a specified financial goal (future value) in 5,10,10,15,20,25 or 30 years. Users must provide their financial goal (the amount of money desired after the specified number of years has elapsed), an interest rate and the length of the investment in years. Create an application that calculates and displays the principal (initial amount to invest) needed to achieve the user's financial goal. Your application should allow the user to invest money for 5, 10,15,20,25 or 30 years. For example, if a customer wants to reach the financial goal of $15,000 over a period of 5 years when the interest rate is 6.6 %, the customer would need to invest $10, 896.96.

Adding the numericUpDown control Place and size the numericUpDown so that if follows the GUI Design Guidelines. Set the numericpDown control's Name property to upYear. Set the numericUpDown control to allow only multiples of five for the number of years. Also,allow the user to select only a duration that is in the specified range of value.

Adding a multiline TextBox Add a TextBox to the Form below the NumericUpDown control. Change the size to 272, 88 and position the TextBox on the Box to display multiple lines and a vertical scrollbar. Also ensure that the user cannot modify the text in the TextBox.

Adding a Click event handler and adding code Add a click event handler for the Calculate Botton. Once in code to the application such that, when the Calculate Button is clicked, the multiline TextBox displays the necessary principal for each five-year interval use the following version of the present-value calculation formula:

$P=a/(1 + r)n$

Where

p is the amount needed to achieve the future value

r is the annual interest rate (for example, .05 is equivalent to 5%)

n is the number of years

a is the future-value amount

PROBLEMA 3

(Array)

Use a one-dimensional array to solve the following problem: A company pays its salespeople on a commission basis. The salespeople receive $200 per week, plus 9% of their gross sale for that week. For example, a salesperson who grosses $5000 in sales in a week receives $200 plus 9% of $5000, or a total of $650. Write a program (using an array of counters) that determines how many of the salespeople earned salaries in each of the following ranges (assume that each salesperson's salary is truncated to an integer amount):

$200-$299
$300-$399
$400-$499
$500-$599
$600-$699
$700-$799
$800-$899
$900-$999
$1000 and over

PROBLEMA 4

(Create and Maintain Telephone Directories)

Write a program to create and maintain telephone directories. Each directory will be a separate sequential file. The following buttons should be available:

Select a directory to access. A list of directories that have been created should be stored in a separate sequential file. When a request is made to open a directory, the list of available directories should be displayed as part of an InputBox prompt requesting the name now listed, the desire to create a new directory created and added to the list of existing directories.

Add name and phone number (as given in the text boxes) to the end of the current directory.

Delete name (as given in the text box) from the current directory

Sort the current directory into name order.

Print out the names and phone numbers contained in the current directory

Terminate the program

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2003

Expertos

InterBay

May 16, 2003

Expert

Program 1

Weighted Binary Trees

What would an ACSL All-Star Contest be without a binary search tree program? Yup, it just wouldn't be an All-Star Contest, so here we go…

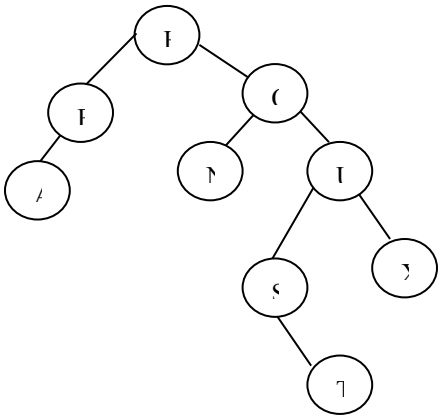
In a binary search tree, the nodes that are near the top of the tree are faster to find that the nodes at the bottom. It would be nice to have the nodes that will be looked for frequently to appear at the top of the tree.

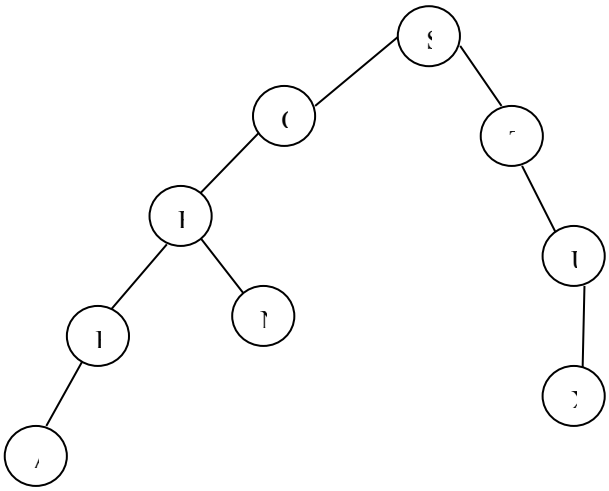
The input in this program is a string. Ignore everything in the string but the letters of the alphabet. Convert all uppercase letters to lowercase (or vice versa); the nodes in the tree will be letters.

Step 1. Build a binary search tree from the input. If you encounter a letter that is already in the tree, stop building the tree. Delete from the input the previous occurrence of this duplicate letter, and change the input so the duplicate letter just encountered is now first in the input. Rebuild the tree from scratch, stopping the next time that a duplicate letter is found (in the modified input), rearranging the input, and so on. Thus, if the input is "CENTER", you'd build a tree with C E N T, stop because there's a duplicate E, rearrange the input to be E C N T and continue building the tree. The keys in the final tree would be E C N T R, in this order.

Step 2. After the tree is built, go through the letters in the original input string and search for each one in the tree. When you find the node, record its depth. The sum of the depths of the nodes for all the input in proportional to the time needed to search in the tree.

For example, consider the input "Houston, Texas" A standard binary search tree (ignoring duplicates) has the tree at the left; the weighted tree that you'll build in the program is at the right. The final time that the weighted tree was rebuilt was when it processed the final S. At that point, the letters were inserted in the order S T O H U N E X A.





The standard binary search tree (left) has a *processing time* of 26 (0+1+2+3+4+1+2+4+1+3+2+3), whereas the weighted version (right) has a *processing time* of 21 (2+1+2+0+1+1+3+1+3+3+4+0).

Sample Data (2 sets of data; the test data has 10 sets of data):

Input	Output
Houston, Texas	21
American Computer Science League (ACSL)	69

Program Weighted Binary Trees

Input	Output
Binary search trees	34
Freedom of speech	27
Dr. Jykell and Mr. Hyde	40
San Francisco	22
California	20
San Francisco, California	47
to whom it may concern:	36
President William Clinton	45
Vice President Al Gore	40
X	0

InterBay

May 16, 2003

Expert

Program 2

Triangle

Given three lines that intersect such that the intersection points form a triangle, find the perimeter of the triangle.

To make the input easy, each line will be specified by two points from among the following 12 points:

A (0,0)	B (2,3)	C (-2,5)
D (-2, -4)	E (1, -2)	F (4, 1)
G (-4, 1)	H (-2, 0)	I (0, -6)
J (3, -4)	K (5, 5)	L (3,0)

Sample Data (sets of data; the test data has 10 sets of data):

Input	Output
DL, GH, HF	13.99079
HD, EJ, FI	23.59552

Program 2, Triangle

Input	Output
AG, GC, CA	13.9804
JL, EJ, EL	9.65685
BC, KL, AE	22.79078
HL, BH, LB	13.16228
EJ, JL, EL	9.65685
ID, FJ, BD	39.37421
HB, BL, LH	13.16228
LE, JE, LJ	9.65685
BF, IL, KG	11.92015
AL, JL, AJ	12.00000

InterBay

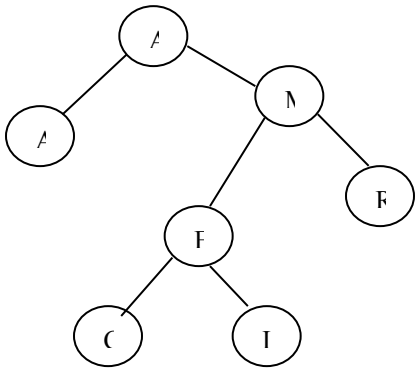
May 16, 2003

Expert

Program 3

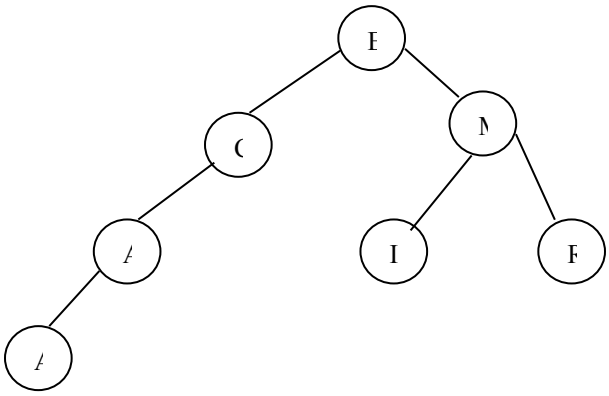
Another Balancing Act

Usually when you build a binary search tree, you add each key to the tree as the key is encountered in the input stream. For example, the binary search tree built from the letters A M E R I C A starting with the A and ending with the final A looks as follows:



If you knew all of the keys in advance, you might choose to add them in the order E A A C M I R to form a perfectly balanced tree. (There are other orders that also yield a balanced tree.) However, determining an order to insert the keys requires that you sort the keys, a time-consuming proposition. This problem explores a method that doesn't involve sorting.

In this program, you'll keep a 2-character look-ahead buffer. That is, in addition to knowing the key you are adding, you also have access to the next two characters. Of the three characters, you should insert the median one. For example, with the letters A M E R I C A, you start off looking at A M E, and you'd insert E as the root. Now, you have A and M in your buffer, and you scan the R. Of these three letters, you'd ass the M. You'd add the I and then the C. Finally, when you scan the final A, you have A and R already in the cache. You insert an A (the median letter), and then you are left with A and R. Insert the smaller of the two first. The final tree looks as follows:



In this problem, you need to compare the shape of a standard binary search tree with one built using a 2-character look-ahead buffer. The input will be 10 sets of data. Each set will consist of a string S. In the string S, ignore everything but the letters of the alphabet; uppercase and lowercase are the same. For each input string,

report how much the internal path length decreases using a 2-character look-ahead buffer. (If the internal path length increases, you'll report a negative number.) For example, AMERICA has an internal path length of 12 using a conventional binary search tree; with the buffer, the path length decreases by 1 to 11.

Sample Input:

Line 1: America
Line 2: North Carolina

Sample Output:

Output 1: 1
Output 2: -4

InterBay

May 16, 2003

Expert

Program 4

ACSLzip

In order for a business to get the best price from the United States Postal Service, the business must bring its mail to the post office in pre-sorted bundles. The bundles must be formed according to the following rules and in the following order:

1. Bundle 10 or more pieces of mail with the same 5 digit zip code and place a "D" sticker on the top piece.
2. Bundle 10 or more pieces with the same first 3 digits of their zip code and place a "3" sticker on the top piece.
3. Bundle 10 or more pieces to the same Area Distribution Center (ADC) and place an "A" sticker on the top piece. An ADC is a way to group together various zip codes. The following table shows the ADC that is used for zip codes whose first 3 digits begin as indicated. For example, the ADC of "028" is used for zip codes that begin 020, 023, 024, 025, …, 029.
4. 105 004, 105-109
5. 117 005, 115, 117-119
6. 106 006-009
7. 110 010-017
8. 021 018, 019, 021, 022, 055
9. 028 020, 023-029
10. Bundle the remaining pieces and place an "MS" sticker on the top piece.
11. No bundle may have more than 125 pieces of mail in it.

There will be ten sets of data. Each set consists of a positive integer N followed by N pairs of numbers that give the number of pieces going to each zip code. For example, the first line below has 8 pieces of mail for 02910 and 6 pieces of mail for 01845. For each set of data, print the number of different types of bundles (D, 3, A, and MS) that you have with labels. You must not list any bundle-types that are zero.

Sample Input:

Line 1: 2, 8, 02910, 6, 01845
Line 2: 8, 8, 02910, 16, 02920, 2, 01824, 6, 01834
9, 01845, 5, 01937, 5, 02244, 15, 02736

Sample Output:

Output 1: MS=1
Output: D=2 3=1 A=1 MS=1

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2003

Principiantes

InterBay

May 16, 2003

Beginners
Progam 1

COUNTCHARS

This problem requires you to perform some analysis on a file of text. More specifically, it requires you to look at each line of the file and report on how many upper-case letters, lower-case letters, and non-letters that the line contains.

The input file will contain at least one line of text, but you do not know how many lines. Lines may vary in length, up to a maximum of 80 characters, and there may be blank lines. Note that the previous two statements imply that a file may contain just a single blank line.

Your program must read the lines of text in the input file and must produce one output line for each line of input in the file. Each such line of output must have the following form:

Line? contains? capital letters,? lower-case letters, and? non-letters.

In which the first question mark (?) represents the line number, the second and third question marks represent the number of upper-case (capital) letters and lower-case (small) letters, respectively, and the final question mark represents the number of non-letter characters. The end-of-line character is itself not counted.

Note carefully the format of each output line: It must be a complete sentence, terminated by a period, and each numerical value is preceded and followed by a single space.

Note in the sample input data files which follow that there are no blank spaces after the last visible character on each line.

==Sample input data contained in COUNTCHARS1.DAT==
This is VERY GOOD TIME, believe it or, not to be QUITE CAREFUL.
If you are not, well, things can go WRONG!!
Not only that, BUT, it will take you time to make them right…!:)
========== Corresponding output for input data from COUNTCHARS1.DAT=
Line 1 contains 25 capital letters, 23 lower-case letters, and 15 non-letters.
Line 2 contains 6 capital letters, 25 lower-case letters, and 12 non-letters.
Line 3 contains 0 capital letters, 0 lower-case letters, and 0 non-letters.
Line 4 contains 4 capital letters, 42 lower-case letters, and 23 non-letters.
========== Sample input data contained in COUNTCHARS2.DAT==
AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz1234567890
~`!@#$%^&*()_-+=|\{}[]:”;’<>?,./ do SEEM be ALL of those punctuation characters.
There once was a man from Peru…
Who dreamt he was eating his shoe!
He awoke in the night

With a terrible fright
And fought it was perfectly true!!

========== Corresponding output for input data from COUNTCHARS2.DAT
Line 1 contains 26 capital letters, 26 lower-case letters, and 10 non-letters.
Line 2 contains 7 capital letters, 32 lower-case letters, and 41 non-letters.
Line3 contains 2 capital letters, 22 lower-case letters, and 10 non-letters.
Line 4 contains 1 capital letter, 26 lower-case letters, and 7 non-letters.
Line 5 contains 1 capital letter, 16 lower-case letters, and 4 non-letters.
Line 6 contains 1 capital letter, 18 lower-case letters, and 3 non-letters.
Line 7 contains 1 capital letter, 25 lower-case letters, and 7 non-letters.
Line 8 contains 0 capital letter, 0 lower-case letter, and 0 non-letters.

InterBay

May 16, 2003

Begginers
Program 2

Decoding an Encoded Textfile

This problem requires you to decode an encoded file of text and display the result (i.e., the “plain text”) on the screen.

First, for simplicity, you may assume that the only characters in any original plain text file were capital letters, the blank space character, and the end-of-line character (also called the “newline character”, or the “RETURN” character”) which indicates where the line breaks are located.

We shall describe how any input file for your program, which must be called TEST.DAT, was encoded. That is, we shall describe how the original plain text file was transformed into the encoded textfile that your program must now decode.

First of all, to encode a file, all characters in it are read one at a time. If the character is one of the capital letters A, B, …, Z then it is encoded by writing out the two digit characters giving its numerical position on the alphabet (01 for A, 02 for B, …26 for Z). If it is an end-of-line character it is encoded as 27 and if it is a blank space it is encoded as 28.

Also, as these digits are written out to the output file, as soon as 40 digits (corresponding to 20 encoded characters) have been written out, a new line of output is started. Thus every line in an encoded file except (usually) the last contains exactly 40 digit characters.

Once decoded, the plain text must be displayed on the screen with line breaks in the proper places, as shown in the sample output below.

```
== Sample input data from DECODE1.DAT ==
2008091928091928200805281521201621202806
1815132801142805140315040504282005192028
0609120527092028031514200109141928190522
0518011228120914051928150628200524202725
1521280301142820080914112815062805010308
2812091405280119280128190514200514030528
0522051428200815210708280920280801192814
1528162114032021012009151427151828251521
2803011428200809141128230801200522051828
0512190528251521281209110527200809192809
192820080528120119202812091405 27

==  End of input data/Start of corresponding output data  ==
THIS IS THE OUTPUT FROM AN ENCODED TEST FILE IT CONTAINS SEVERAL LINES OF TEXT
YOU CAN THINK OF EACH LINE AS A SENTENCE EVEN THOUGH IT HAS NO PUNCTUATON OR
YOU CAN THINK WHATEVER ELSE YOU LIKE THIS IS THE LAST LINE
====  End of output data  ===
```

==== Sample input data from DECODE2.DAT
==========
2728282828200805190528202315281209140519
2806151212152328012806091819202802120114
1128120914052728282828011404280118052805
0103082809140405142005042802252806152118
2819160103051927
==== End of input data/Start of corresponding output data
==========

THESE TWO LINES FOLLOW A FIRST BLANK LINE AND ARE EACH INDENTED BY FOUR SPACES
==== End of output data ====

InterBay
May 16, 2003
Beginners
Program 3

Virus Detection

When examining a file, a certain virus scanner looks for, among other things, occurrences of the word “virus” in the file. For example, the line

ab12$virus23

appearing in some file would be marked as suspect, as it contains an occurrence of the string “virus”. It is safer, however, to also look for what are called “scattered occurrences” of the string “virus”. For example, the line

contains a “scattered occurrence” of the string “virus”, which is indicated by the underline characters. Note that even in a “scattered occurrence” of the word the order of the letters remains the same as in the correctly spelled word “virus”.

Your task is to write a program that reads each line in a textfile called TEST.DAT, and outputs the line numbers of every line of text which contains either a direct occurrence of the string “virus”, or a “scattered occurrence” of the string “virus”, as described above. The first line of the file contains just the number of lines of text to follow. The lines of text in the file from line 2 of the file onward have “line numbers” starting at 1.

====Sample input data from VIRUS1.DAT
4
dsffdvi5wqdhdr
vvvvirus23e4
@#1vviwdscr32uu2ss
%%gfsd

====End of input data/Start of corresponding output data====
2
3
====End of output data====

====Sample input data from VIRUS2.DAT====
6
itisveryinvigoratingtorunaroundtheskyscraper.
This line does not contain the v*i*r*u*s string. Or does it?
)(^*%&V#@$@¡*)*)&R879078u*)*)*S
Nowisthetimeforallhackersandvvvviiirrruuussswriterstoceaseanddesis
T!
Fajlcvjioryweoprhavkahyiesdjvoefhbg
Piroutrgbsfoguofgnktnhnskhkhkhkhkhbfskkklhfeikbmadfgiyewt
====End of input data/Start of corresponding output data====
1
2
4
====End of output data====

InterBay

May 16, 2003

Beginners
Program 4

Number Properties

A positive integer is considered prime if it is evenly divisible only by itself and 1. Also, by convention, 1 is not itself a prime. Thus, the sequence of primes begin: 2, 3,5,7,11,13,17,…

A positive integer is called perfect if it is the sum of its proper divisor. For example, 28 is a perfect number because 28= 1+2+4+7+14.

Your assignment is to write a program that will input a sequence of integers, one per line, and output the properties of each integer in the following format:

If the number is neither prime nor perfect, output "Dull".
If the number is prime but not perfect, output "Prime".
If the number is perfect but not prime, output "Perfect".
If the number is both prime and perfect, output "This program doesn't work".
All input will be positive integers. The end of input will be marked by the number 0 (no output should be generated for 0).

==========
Example:
====**Input**====
28
7
140
1
5
0
====**Output**====
Perfect
Prime
Dull
Dull
Prime

InterBay

May 16, 2003

Beginners
Program 5

Time Card

Jenny just started work as a programmer for Justine's Java Workshop. She is paid $10 an hour, with a few exceptions. She earns an extra $1.50 an hour for any part of a day where she works more that 8 hour, and an extra $2.50 an hour for hour beyond 40 in any one week. Also, she earns a 25% bonus for working on Saturday, and a 50% bonus for working on Sunday. The bonuses for Saturday and Sunday are computed based on the hours worked those days; they are not used to calculate any bonus for working more than 40 hours in a week.

You'll be given the number of hours Jenny worked each day in a week (Sunday, Monday, …, Saturday), and you need to compute her salary for the week. The input will be positive integers, less than or equal to 24. The output must be formatted with a dollar sign and rounded t the nearest penny. For example, "$2" and "$2.136666" are wrong answers; the correct versions are "2.00" and "$2.14", respectively. There may not be any embedded spaces in your answers. There will be 5 sets of data.

====Sample Input:====

Line 1:0, 8, 8, 8, 10, 6, 0
Line 2:4, 0, 0, 0, 0, 6, 0
==========

Sample Output:

Output 1: $403.00
Output 2: $120.00

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2002

Expertos

Universidad Interamericana de Puerto Rico
Recinto de Bayamón
Terceras Olimpiadas de Programación
INTERBAY 2002

CATEGORÍA DE EXPERTO

Instrucciones generales:

Todos los problemas de esta categoría serán interactivos. (NO leen datos desde archivos externos). Puedes utilizar para resolverlos los siguientes lenguajes: Visual Basic o C++. Deben entregarse en disco tan pronto como lo hayas terminado.

PROBLEMA 1

Multiplicación Rusa

En el siglo pasado en Rusia, los campesinos se servían de un ingenioso método de multiplicación para el que sólo necesitaban saber encontrar el doble y la mitad de un número, así como la elemental operación de sumar. Con este sistema había que formar dos columnas encabezadas por las cifras que se iban a multiplicar.

El método opera en la siguiente forma:

Ejemplo: Multiplíquese 97 por 39

a) Se saca sucesivamente la mitad a los números situados en la columna de la izquierda (sin tomar en cuenta los residuos), y el doble a los de la columna derecha. Se continúa esta operación hasta que la columna izquierda se reduzca a la unidad.

97	x	39
48	x	78
24	x	156
12	x	312
6	x	624
3	x	1248
1	x	2496

b) Se tachan todas las cifras pares que figuren en la columna izquierda y por consiguiente las cifras correspondientes en la columna derecha.

9	X	3
7		9
4	X	7
8		8
2	X	1
4		56
1	X	3
2		12
6	X	6
		24
3	X	1
		248
1	X	2
		496

c) Finalmente se suman las cifras que quedan en la columna derecha. El total de la suma es el resultado.

39 + 1248 + 2496 = 3783

Problema:

Desarrolle un programa que solicite al usuario y multiplique dos cifras no menores de 2 dígitos ni mayores de 5. El programa tiene que efectuar los mismos pasos mencionados anteriormente para poder llegar al resultado.

Ejemplo de Input:

Indique el primer dígito: **97**
Indique el segundo dígito:**39**

Ejemplo de Output:

97 x 39
48 x 78 <-Se elimina
24 x 156 <-Se elimina
12 x 312 <-Se elimina
6 x 624 <- Se elimina
3 x 1248
1 x 2496

39 + 1248 + 2496

RESULTADO= 3783

PROBLEMA 2

Vertical Histogram

Escribe un programa que acepte dígitos (0-9) como Input. De como resultado el histograma vertical representativo de cada dígito. Calcula también la suma de todos los dígitos, el promedio, y da como resultado la lista de los números por debajo del promedio sin repetir y la lista de los números por debajo del promedio sin repetir y la lista de los números por encima del promedio sin repetir.

Verifica tu programa con este set de 13 dígitos.
1,7,2,9,6,7,1,3,7,5,7,9,0

Ejemplo de Input:

Enter a Number :12
Enter 12 digits:
1,7,2,9,6,7,1,3,7,5,7,9

Ejemplo de Output:

```
        *
        *
*       *         *
***   ***         *
0123456789
```

La suma es : 64 El promedio es : 5.3
Los números por debajo son: 1,2,3
Los números por encima: 6,7,9

PROBLEMA 3

Write a program to read in two 2-dimensional arrays, and then multiply one by the other. This called matrix multiplication. For example, if first Array (2-row by 2-column array) appears as

7
3

and secondArray (2-row by 1-column array) appears as

8
6

then the product matrix is

productMatrix[0] [0] = 2 * 8 + 7 * 6
productMatrix[1] [0] = 9 * 8 + 3 * 6

Matrix multiplication can be done only if the number of columns in the multiplicand (the first array) equals the number of rows in the multiplier (the second array).

The program should read in the two arrays, test to see of multiplication is possible, and then multiply them if it is. The output will be a printout of the two arrays and will either output the product array or print a message saying that multiplication in now possible.

PROBLEMA 4

Write an interactive program that plays tic-tac-toe. Represent the board as a three-by-three character array. Initialize the array to blanks and each player in turn to input a position. The first player's position will be market on the board with a 0, and the second player's position will be market with an X. Continue the process until a player wins or the game is a draw. To win, a player must have three marks in a row, in a column, or on a diagonal. A draw occurs when the board is full and no one has won.

Each player's position should be input as indices into the tic-tac-toe board that is, a row number, a space, and a column number. Make the program user-friendly.

After each game, print out a diagram of the board showing the ending position. Keep a count of the number of games each player has won and the number of draws. Before the beginning of each game, ask each player if he or she wishes to continue. If either player wishes to quit, print out the statistic and stop.

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2002

Intermedios

Universidad Interamericana de Puerto Rico
Recinto Bayamón
Primeras Olimpiadas de Programación
INTERBAY 2002

CATEGORÍAS DE INTERMEDIOS

Instrucciones generales:

Todos los problemas de esta categoría serán interactivos. (NO leen datos desde archivos externos). Puedes utilizar para resolver los siguientes lenguajes: Visual Basic o C++. Deben entregarse en disco tan pronto lo hayas terminado.

PROBLEMA 1

The In between Sum

Problema:

You are given two numbers, A and B, and you have to find the total of all the numbers between these two numbers. For example if you are given the numbers 9, and 15, you 15, you must find the sum 10+11+12+13+14 which is 60.

Ejemplo de Input:

The input is the pair of intergers, *A* and *B* (1 <=A <=B <= 30000).

Ejemplo de Output:

The output will be the sum of all the numbers between A and B.

Example 1

Input
9 15
Output
60

Example 2

Input
10 20
Output
135

PROBLEMA 2

Usando una tabla sencilla (un arreglo de orden 1) resuelva el siguiente problema. Una compañía le paga a sus vendedores reciben $200 semanalmente más el 9% de las ventas de esa semana. Por ejemplo, un vendedor que venda $5000, recibe $200 más el 9% de $5,000, ó $650.

Usando un arreglo de contadores, determine cuántos vendedores recibieron un salario en cada una de las siguientes escalas:

a. $200-299-2
b. $300-399-4
c. $400-499-4
d. $500-599-3
e. $600-699-4
f. $700-799-4
g.$800-899-3
h.$900-999-4
i. $1,000+ -7

Usa los siguientes datos de ventas:

1000	1100	1400	1700	1900	2100	2400
2700	3000	3300	3600	3900	4200	4500
4700	5000	5300	5600	5900	6200	6500
7000	7200	7500	7800	8100	8300	8700
9000	9200	9500	10200	10300	11000	13000

PROBLEMA 3

Escriba un programa que lea un carácter de “A” a la “Z” y produzca una salida con forma de pirámide que consiste de los caracteres entrados. El carácter superior debe ser la letra “A” y en cada nivel subsiguiente la letra entrada debe caer en el medio de los caracteres entrados anterirmente.

A
ABA
ABCBA
ABCDCBA
ABCDEDCB

PROBLEMA 4

(Print a String Backward)

Escriba una función recursiva que reciba un arreglo que contiene un “string” como argumento, escrbe el “string” y devuelve nada. El programa termina su ejecución cuando encuentra un “null carácter”.

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2002

Principiantes

Universidad Interamericana de Puerto Rico
Recinto de Bayamón
Primeras Olimpiadas de Programación
INTERBAY 2002

CATEGORÍAS DE PRINCIPIANTES

Intrucciones generales:

Todos los problemas de esta categoría serán interactivos. (NO leen datos desde archivos externos). Puedes utilizar para resolverlos los siguientes lenguajes: Visual Basic o C++. Deben entregarse en disco tan pronto como lo hayas terminado.

PROBLEMA 1

Prepara un programa que cuenta el número de letras, puntos y los signos de exclamación en los primeros 50 caracteres de un "string". Imprime el "string" y el total de letras, puntos, y signos de exclamación.

Input: ¡HOLA!. Fue un placer conocerte. ¡Que te valla bien!

Output: ¡Hola!. Fue un placer conocerte. ¡que vallas bien!
Letras= 37 Exclamación =4 Puntos=2

PROBLEMA 2

Rotating Words

You are required to rotate a word a certain amount. For example, to rotate the word “Computer” by 1 results in “rCompute”. Rotating it two more times gives you “terCompu”.

Ejemplo de Input:

The first line contains the word (which will not have more than 15 letters). The second line contains an integer n, which will be less than the length of the word. You must rotate the word n times.

Ejemplo de Output:

The output will be the rotated word.

Example 1

Input
Entre la palabra: **Computer**
Entre la cantidad: **3**
Output
terCompu

Example 2

Input
Entre la palabra: **Program**
Entre la cantidad: **1**
Output
mProgra

PROBLEMA

A parking garage changes a $6.00 minimun fee to park for up to three hours. The garage charges an additional $1.50 per hour for each hour or part there of in excess of tree hours. The maximum charge for any given 24-hours at a time. Write a program that will calculate and print the parking charges for each customer who parked his or her car in this garage yesterday. You should enter the hours parked for each customer. Your program should print the results in a neat tabular and should calculate and print the total of yesterday's receipts. The program should use the procedure **CalculateCharges** to determine the charge for each customer.

PROBLEMA 4

Write a program that converts letters of the alphabet into their corresponding digits on the telephone. The program should let the user enter letters repeatedly until a Q or a Z is entered. (Q and Z are the two letters that are not on the telephone). An error message should be printed for any nonalphabetic character that is entered.

The letters and digits on the telephone have the following correspondence.

ABC=2	JKL=5	TUV=8	DEF=3
MNO=6	WXY=9	GHI=4	PRS=7

Here a letter: P
The letter P corresponds to 7 on the telephone
Enter a letter: A
The Letter A corresponds to 2 on the telephone
Enter a letter: D
The letter D corresponds to 3 on the telephone
Enter a letter: 2
Invalid letter. Enter Q or Z to quit
Enter a letter: Z
Quit

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2001

Expertos

Universidad Interamericana de Puerto Rico
Primeras Olimpiadas de Programación
InterBay 2001

CATEGORÍA EXPERTOS

Instrucciones generales:

Todos los problemas de esta categoría serán interactivos y con archivos externos. Puedes utilizar para resolverlos los siguientes lenguajes: Visual Basic o C++. Deben entregarse en disco tan pronto como lo hayas terminado.

PROBLEMA 1

Un *"palindrome"* es una palabra que se deletrea de igual manera al derecho o al revés, tal como "radar". Escribe un programa que acepta una serie de caracteres ("string") con un punto al final y determina si la palabra (sin incluir el punto) es un "*palindrome"*. Tu puedes asumir que la entrada de la palabra será en minúscula y que tiene un largo máximo de 20 caracteres. El programa no tiene que cotejar si la palabra existe realmente el español o inglés. Por ejemplo, la palabra "aabbcbbaa" se considerará un "*palindrome"* por tu programa.

Permite la iteración del programa de manera que le permita al usuario cotejar palabras adicionales hasta que decida terminar.

PROBLEMA 2

Escribe un programa que le asigne asientos a los pasajeros. Asume que los asientos del pequeño avión se asignan siguiendo el patrón a continuación.

1	A	B		C	D
2	A	B		C	D
3	A	B		C	D
4	A	B		C	D
5	A	B		C	D
6	A	B		C	D
7	A	B		C	D

El programa debe desplegar los asientos del avión, marcando con una"**X**", aquellos asientos ocupados. Por ejemplo después que los asientos **1A, 2B y 4C** se ocupan, se debe desplegar lo siguiente:

1	X	B		C	D
2	A	X		C	D
3	A	B		C	D
4	A	B		X	D
5	A	B		C	D
6	A	B		C	D
7	A	B		C	D

Después de desplegar los asientos disponibles, el programa debe pedir al asiento deseado. El usuario entra el número del asiento y se despliega los datos actualizados. Esto continua hasta que todos los asientos estén ocupados o hasta que el usuario indique que desea terminar. Si el usuario pide un asiento ocupado, el programa debe así indicarlo y pedir que se entre otro asiento.

PROBLEMA 3

Escribe una función que llamaremos ***merge-lists*** que acepta dos argumentos "call-by-reference" que son variables que apuntan al principio de dos listas encadenadas ("linked lists") de tipo ***int.*** Asume que las dos listas encadenadas están ordenadas de manera que el valor al principio de la lista es el valor más pequeño que apunta al número que le sigue en tamaño y así sucesivamente. La función devuelve el valor del apuntador de una lista encadenada que consiste de las dos listas anteriores. Los valores de esta lista también están ordenados. Esta función no creara ni destruirá los nodos de la lista. Al terminar la función, los argumentos deben tener como contenido el valor de NULL.

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2001

Intermedios

Universidad Interamericana de Puerto Rico
Primeras Olimpiadas de Programación
InterBay 2001

CATEGORÍA DE INTERMEDIOS

Instrucciones generales:

Todos los problemas de esta categoría serán interactivos y con archivos externos. Puedes utilizar para resolverlos los siguientes lenguajes. Visual Basic o C ++. Deben entregarse en disco tan pronto como lo hayas terminado.

PROBLEMA 1

Programa Planilla forma corta.

Este programa calculará la contribución a pagar, o si está exento de pagar o si le tienen que devolver de la cantidad retenida. El programa pedirá los siguientes valores:

a) Nombre del contribuyente
b) Cantidad ganada en el año
c) Cantidad retenida durante el año
d) Deducciones
    1) Cantidad de dependientes (se multiplica por 1,200)
    2) Intereses de auto (máximo de 1200, verificar dicha cantidad)
    3) Estatus. Soltero o Casado (deducción fija Soltero =3,300. Casado =6,000)
    4) Gastos ordinarios necesarios (máximo de 3% de la cantidad ganada, verificar cantidad)
    5) Cantidad invertida en IRA (si es soltero máximo de 3,000 y Casado 6,000)
e) Cálculos
    1) Calcular en una función la cantidad total de deducciones (sumar las deducciones aplicables)
    2) Calcular en una función la contribución a pagar utilizando la tabla de contribución, si está exento (si la cantidad ganada después de las deducciones es menor o igual a cero) o si le deben resolver de la cantidad retenida y cuánto.
    3) El programa deber dar los resultados correspondientes y un mensaje para volver al Menú o a comenzar con un nuevo contribuyente.

Tabla Contributiva

No mayor de $2,000	**7.5%**
En exceso de $2,000 pero no en exceso de $17,000	**$150 más el 11% del exceso de $2,000**
En exceso de $17,000 pero no en exceso de $30,000	**$1,800 más el 16.5% del exceso de $17,000**
En exceso de $30,000 pero no en exceso de $50,000	**$3,945 más el 29.5% del exceso de $30,000**
En exceso de $50,000	**$9,845 más el 33% del exceso de $50,000**

PROBLEMA 2

En un archivo externo de tipo TEXT existe una lista de números de teléfonos. Éstos contienen números y letras eje: Kal-2854 ó 754-Exam. Hacer una lista de los números de teléfonos originales y al lado el equivalente en formato de sólo número o sea remplazar las letras por números según el código del teléfono. Debe contener por lo menos 15 record.

Código

2=A-B-C 3=D-E-F 4=G-H-I 5-J-K-L 6=M-N-O
7=P-Q-R-S 8=T-U-V 9=W-X-Y-Z

PROBLEMA 3

2. Hacer un programa que pida escribir un párrafo y luego des ENTER para terminar. Da como resultado la cantidad de caracteres usados sin contar los espacios y la cantidad de palabras que contiene.

PROBLEMA 4

Programa Promedios.

Leer una lista de 15 notas de exámenes de un máximo de 100 que está en un archivo externo llamado **notas.dat** y guárdalos en un Array. En una función se calculará el promedio de dicha lista. Dar como resultado:

a) La lista de los números
b) El promedio de dicha lista
c) Una lista de las notas por debajo del promedio
d) El promedio de la lista de notas por debajo
e) Una lista de las notas por encima del promedio
f) El promedio de la lista por encima del promedio

La función que calcula el promedio se llamará 3 veces.

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2001

Principiantes

Universidad Interamericana de Puerto Rico
Primeras Olimpiadas de Programación
InterBay 2001

CATEGORÍA DE PRINCIPIANTES

Instrucciones generales:

Todos los problemas de esta categoría serán interactivos o de archivos externos. Puedes utilizar para resolverlos los siguientes lenguajes: Visual Basic o C++. Deben entregarse en disco tan pronto como lo hayas terminado.

PROBLEMA 1

Programa de números

Hacer un programa que pida una cantidad de números indeterminados y un sentinel para terminar. Dar como resultado lo siguiente:

a) Cuántos fueron positivos
b) Cuantos fueron negativos
c) Cuantos fueron ceros
d) Promedio de los positivos
e) Lista de los positivos
f) Lista de los negativos
g) Cantidad total de números entrados sin contar el sentinel

PROBLEMA 2

Programa para sortear por edad

Desarrolle una aplicación que lea de un **archivo** los siguientes campos número seguro social, apellido paterno, nombre, pueblo y fecha de nacimiento. Generan un reporte que presente todos los campos y calcule la edad de la persona. El reporte se va a imprimir en orden de edad de menor a mayor.

Datos del Archivo

111111111, Rodríguez, Marisol, Bayamón, 25 de diciembre de 1980
222222222, Martínez, Edgar, Dorado, 12 de enero de 1962
333333333, González, Juan, Dorado, 12 de octubre de 1983
444444444, Rodríguez, Iván, Vega Baja, 2 de febrero de 1962
555555555, Alomar, Roberto, Salinas, 16 de junio de 1967

PROBLEMA 3

Programa para calcular cuentas a cobrar

La compañía XYZ tiene problemas con las cuentas a cobrar y lo contrata para desarrollar una aplicación que le diga los clientes que cantidad de dinero deben a 30 días, a 60 días y sobre 90 días. Tienen que leer número de factura, nombre de cliente, fecha factura y total de la factura.

El reporte tiene que estar organizado por cliente y tiene que informar la cantidad que debe a 30 días a 60 días y sobre 90 días.

Datos del archivo

33433a, Compañía de Energía, 27 de marzo de 2001, 4353.56
43536b, El Vocero, 26 de marzo de 2001, 3423.87
43563j, Microsoft, 6 de abril de 2001, 4358.34

43532h, Compañía de Energía, 27 de marzo de 2001,6534.25
65353h, Compañía de Energía, 25 de abril de 2001,2345.78
635351, El Vocero, 1 de mayo de 2001, 4356.56

PROBLEMA 4

Programa para recibo de venta

Una cajera necesita saber el cambio de una compra de un producto. Ella va a entrar los nombres de los productos y sus repetidos precios de venta. Luego que sume el total de la venta va a entrar la cantidad recibida por el cliente. La aplicación tiene que informar el cambio que se le va a dar al cliente tanto en moneda como en dólares. Ejemplo: si la venta total es $12.10 y el cliente da un billete de $20, el cambio va a ser un billete de $5, dos billetes de $1, tres pesetas, un vellón de 10 centavos y un vellón de 5 centavos. Además del total de la vuelta, diseñe el recibo como si fuera una tienda por departamentos.

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2001

Premiaciones

InterAmerican University of Puerto Rico
Bayamón Campus
Informatic and Telecommunications Department
Informatic Student Association (ISA/AEI)

COMPUTER PROGAMMING CHALLENGE II 2001

CATEGORY: ADVANCED

FIRST PLACE
TEAM NAME: Los Mofongos
PARTICIPANTS NAMES: Juan C. Vélez, Juan Pablo León
EMAIL:
1 VISUAL STUDIO ENTERPRISE EDITION
1 WINDOWS 2000 PROFESSIONAL

SECOND PLACE
TEAM NAME: Las Girls Scout
PARTICIPANTS NAMES: Ismael Placa, María del Mar Álvarez
EMAIL:
1 VISUAL STUDIO PROFESSIONAL EDITION
1 IBM WEB SPHERE STUDIO

THIRD PLACE
TEAM NAME: bit factory 10
PARTICIPANTS NAMES: Eurípides Rivera, Carlos Reyes
EMAIL: euri_rivera@hotmail.com, sonic@coqui.net
1 VISUAL STUDIO PROFESSIONAL EDITION
1 VISUAL AGE SMALL TALK ENTERPRISE

CATEGORY: INTERMEDIATE

FIRST PLACE
TEAM NAME: Los Marroneros
PARTICIPANTS NAMES: George González, Luis Carmoega
EMAIL:
1 VISUAL STUDIO ENTERPRISE EDITION
1 VISUAL AGE FOR JAVA

SECOND PLACE
TEAM NAME: Cyber Spiderman
PARTICIPANTS NAMES: Hao Wei Wu, Daniel Díaz
EMAIL:
1 VISUAL STUDIO PROFESSIONAL EDITION
1 BORLAND DELPHI DEVELOPMENT TOOLS

THIRD PLACE

TEAM NAME: Temibles
PARTICIPANTS NAMES: Elsie Malavé, Myrna Irizarry
EMAIL:lisi80@hotmail.com
1 VISUAL STUDIO PROFESSIONAL EDITION
1 VISUAL BASIC 6.0 DEVELOPMENT (BOOK)

CATEGORY: BEGINNERS

FIRST PLACE
TEAM NAME: Reales para Cristo
PARTICIPANTS NAMES: Ángel Colón, María Marrero
EMAIL: colon_angel@hotmail.com, mmarerro3881@bc.inter.edu
1 VISUAL STUDIO ENTERPRISE EDITION
1 OFFICE 2000 DEVELOPER

SECOND PLACE
TEAM NAME: No Code
PARTICIPANTS NAMES: Wilson Díaz, Martin Reyes
EMAIL: yield@prtc.net, reyesmartin18@hotmail.com
1 VISUAL STUDIO PROFESSIONAL EDITION
1 THE COMPLETE VISUAL BASIC 6.0 TRAINING COURSE

THIRD PLACE
TEAM NAME: @!/3$5^&..>
PARTICIPANTS NAMES: Mayra Vázquez, Jonathan Nazario
EMAIL: mayrita17@hotmail.com, joscarin@caribe.net
1 VISUAL STUDIO PROFESSIONAL EDITION
1 WEB APPLICATION DEVELOPMENT USING MS VISUAL INTERDEV 6.0 (BOOK)

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2000

Expertos

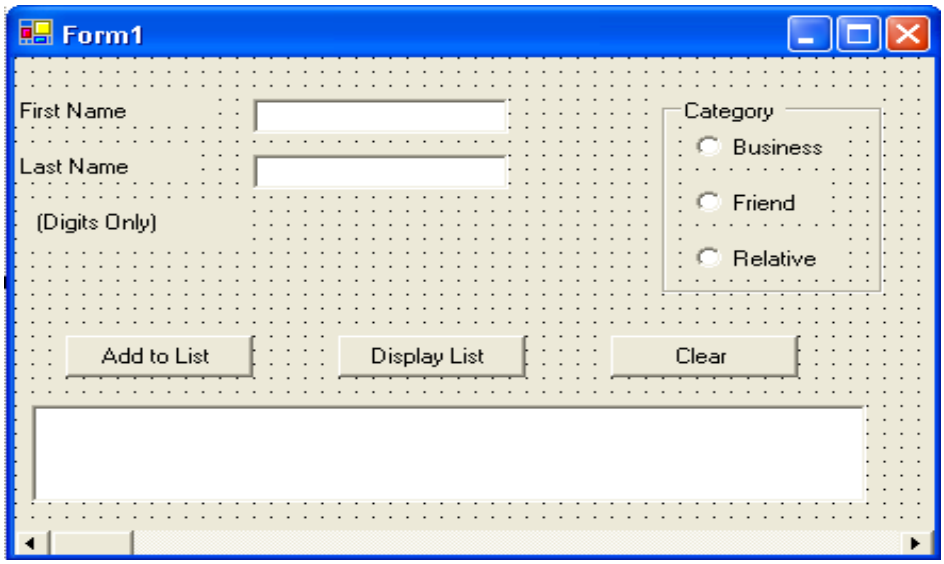
Universidad Interamericana de Puerto Rico
Primeras Olimpiadas de Programación
INTERBAY 2000

CATEGORÍA DE EXPERTO

Instrucciones generales:

Todos los problemas de esta categoría serán interactivos. (NO leen datos desde archivos externos). Puedes utilizar para resolverlos los siguientes lenguajes: Visual Basic o C++. Deben entregarse en disco tan pronto como lo hayas terminado.

1) Code a program that enables a person to input names, phone numbers, and categories (Friend, Relative, Business) of acquaintances (maximum of 25). Input the first names, last names, and phone numbers into text boxes. Use option buttons for the category. When the user clicks an Add to List button, the program stores the name, number, and category in array, displays how many names are saved, and clears the input values from the screen. A message label is used for error messages and to indicate how many names have been added to the arrays. When the user clicks the clear button, the text boxes, message, category, and list box clear. When the user clicks the display list button, all the values from the arrays are displayed in the list box at the bottom. Display a special message if Display list is clicked before any names are in the arrays. A rough layout of the form is shown here.



2) A life insurance company has hired you to write a program to print a list of their customers and the premium that each customer pays. Premiums are based in the age customer was when he or she became a customer. The following table is used to determine each customer's premium, but these rates are subject to change.

Age	Premium
25	$277.00
35	287.00
45	307.00
55	327.00
65	357.00

Each age listed in this table is the upper limit for the premium. For example, if a customer signed up for a policy when she was 37, she would pay $307.

Write a program that reads the list of sequential file into parallel arrays, then reads in the customer's names and ages when they bought the policies into another pair of parallel arrays. The table and the customer's names and ages are stored in two files. Print out a formatted, labeled list showing each customer's name, his or her age when the policy was bought, and the customer's premium.

3) Consider the sequence of digits from 1 trough N (where N=9) in increasing order 1 2 3 4 5 …N and insert either a (+) for addition or a (-) for subtraction or a () [blank] to run the digits together. Now sum the result and see id you get zero.

Write a program that will find all sequence of length N that produce a ZERO SUM.

TEST CASE

Input
7
Output
1+2-3+4-5-6+7=0
1+2-3-4+5+6-7=0
1-2+3+4-5+6-7=0
1-2-3-4-5+6+7=0
1-23+4+5+6+7=0
1-23-45+67=0

You may test this program by entering the integer from the keyboard.

4)A teacher maintains a random-access file containing the following information for each student: name, social security number, grades on each of two hourly exams, and the final exam grade. Assume the random-access file GRADES.TXT has been created with string fields of lengths 25 and 11 and three numeric fields, and all the names and social security numbers have been entered. The numeric fields have been initialized with zeros. Write a program with the five command buttons "Display First Student", "Record Grade(s) & Display Next Student", "Locate Student", "Print Grade List", and "Done" to allow the teacher to do the following.

(a) Enter all grades for a specific exam.

(b) Locate and display the record for a specific student so that one or more grads may be changed.
(c) Print a list of final grades that can be posted. The list should show the last four digits of the social security number, the grade on the final exam, and the semester average of each student. The semester average is determined by the formula (exam1 + exam2 + 2 * final exam)/4.

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2000

Intermedios

Universidad Interamericana de Puerto Rico
Primeras Olimpiadas de Programación
INTERBAY 2000

CATEGORÍAS DE INTERMEDIOS

Intrucciones generales:

Todos los problemas de esta categoría serán interactivos. (NO leen datos desde archivos externos). Puedes utilizar para resolverlos los siguientes lenguajes: Visual Basic o C++. Deben entregarse en dusco tan pronto como lo hayas terminado.

PROBLEMA 1

Escribe un programa que pida una cantidad de dinero. Dar como resultado el número de pesetas, dimes, vellones y centavos que al sumarlos de la cantidad de dinero entrado. Debe ser el número mínimo de menudo.
Ejemplo: cantidad entrada $2.17
Resultado:

8 peseta(s)
1 dime (s)
1 vellón (es)
2 centavo (s)

PROBLEMA 2

Escribe un programa que implemente exponenciación. El programa pedirá valor para N que será la base y valor para E que será el exponente de N, ambos tienen que ser valores enteros.
Debe seguir las siguientes reglas:

Cuando E>0=(N * N * …..N) E veces
Cuando E=0=1
Cuando E<0=1/ (N * N * ….N) E veces

*No puede usar ninguna función pre-definida

Ejemplo: número entrado: 5 Exponente entrado: 3
Resultado:

Resultado de N elevado a la E es:125

PROBLEMA 3

Escribir un programa que pida la dimensión cuadrada para formar una matriz. La matriz se creará con números al azar entre 0 y 9. Pondrá en pantalla la matriz creada y al lado otra matriz que estará multiplicada por 3.
Ejemplo: Número entrado es: 3 (cero para terminar)

Matriz			Matriz por 2		
3	1	4	6	2	8
2	9	2	4	18	4
0	3	7	0	6	14

PROBLEMA 4

Escribir un programa de reservación de asientos para un avión. El avión sólo tiene 10 asientos con una sección de FUMAR y otra de NO FUMAR.
Pedir con Input el nombre del pasajero y la sección deseada.
*Los asientos del 1-5 es sección de FUMAR
*Los asientos del 6-10 es sección de NOFUMAR
* 0 se le asigna al asiento al inicio para indicar que está desocupado y 1 cuando es reservado.
*No puede reservar asientos que ya estén reservados
*Cuando el pasajero desee sección de FUMAR y este llena, pero la de NO FUMAR todavía hay asientos, preguntarle si desea la otra sección y viceversa
*Cuando todos los asientos estén ocupados dar un mensaje de que el vuelo está lleno.
***Dar como resultado cada vez que hayan reservaciones, el nombre, número de asiento y sección (FUMAR, NO FUMAR)
*** El programa debe correr hasta que se llenen todos los asientos.

Universidad Interamericana de Puerto Rico
Recinto de Bayamón

COMPETENCIAS DE PROGRAMACIÓN 2000

Principiantes

Universidad Interamericana de Puerto Rico

Primeras Olimpiadas de Programación

INTERBAY 2000

CATEGORÍA DE PRINCIPIANTES

Instrucciones generales:

Todos los problemas de esta categoría serán interactivos. (NO leen datos desde archivos externos). Puedes utilizar para resolverlos los siguientes lenguajes: Visual Basic o C++. Deben entregarse en disco tan pronto como lo hayas terminado.

PROBLEMA 1

Escribe un programa que ponga en pantalla un triángulo símbolos. El número de líneas en el triángulo y el símbolo será entrado por el teclado. Debe asegurarse que el número sea válido o sea que sea mayor de 1 y menor de 25. De no ser válido dar un mensaje de error y que vuelva a preguntar hasta que sea válido.
Por ejemplo si se entra 5 y '*' la salida será:

```
    *
   ***
  *****
 *******
*********
```

PROBLEMA 2

Escribe un programa que calcule el interés que ganará de una cantidad de dinero invertida. Debe escoger de un menú o botones la cantidad de años que será entre 1 y 10. También escogerá de un menú o botones el por ciento de interés aplicable que será entre un 5% y un 15%. Debe repetir los cálculos hasta que se escoja salir del programa.

Ejemplo: cantidad invertida $1000 a 2 años a un 5% el resultado será: $1200

PROBLEMA 3

Escribe un programa que simule una cuenta de débito (ATH). Debe poner como constante el balance actual y el password que será numérico. Al comenzar la corrida del programa debe pedirle el password y permitirle intentarlo solamente 3 veces, de no entrarlo correcto terminar el programa. Si el password es correcto saldrá un menú con las siguientes alternativas:

1. **Retiro-** pedir la cantidad y dar como resultado el balance hasta el momento. Debe verificar que no se sobre gire la cuenta de ser así dar mensaje: "no tiene fondos suficientes" y el balance.
2. **Depósito-** pedir la cantidad de depósito y dar como resultado el balance disponible.
3. **Balance-** Dar como resultado el balance.

Luego de los resultados debe preguntar si desea hacer otra transacción o desea salir.

PROBLEMA 4

Escribir un programa que pida 2 números. (positivos, negativo o cero). Dar como resultado los siguientes mensajes "resultado será positivo", "resultado será negativo" o "resultado será cero" para las operaciones matemáticas de multiplicación y suma. NO PODRÁ SUMAR NIMULTIPLICAR LOS NÚMEROS.

Ejemplo: Entre un primer número: **-20**
Entre segundo número:**20**

Multiplicación: resultado será negativo
Suma: resultado será cero

MISCELÁNEOS

Fecha: 04/15/2000	**Nombre de la competencia**: _____
Categoría: Experto	**Universidad**: UPR BAYAMON .
Autor: Nelliud Torres	**Tipo de competencia**: Eliminatorias .
Problema #: _____	**Algoritmos**: _____

COBOL DESCRIPTION GENERATOR

Source File Name: COBOGE.XXX

Input File Name: GEN-I.LIB

Output File Name: GEN-O.LIB

En un centro de computos crearon un editor que diseña un reporte por pantalla y una vez creado genere el código de COBOL para la descripción del archivo. Tu tarea es crear el módulo que recoge el archivo con el diseño de la pantalla y generar las instrucciones de COBOL en otro archivo.

Ejemplo:

ARCHIVO CON EL FORMATO DEL REPORTE	DESCRIPCION DEL REPORTE

```
H    UNIVERSIDAD DE PUERTO RICO
H        RECINTO DE BAYAMON
H       REGISTRO DE CHEQUES
H
HFECHA: <MM/DD/YY>        PAGE: <ZZ9>
H
HACCOUNT  CHECKS  CHECK           CHECKS
H NUMBER  BEFORE  NUMBER CHK AMT   AFTER
D<999999>  <99>  <ZZZZ9><Z,ZZZ.99> <99>
T                      <ZZZ,ZZZ.99>
```

```
01 HEADING-1.
   05        PIC X(5).

   05         PIC X(26) VALUE
      "UNIVERSIDAD  DE  PUERTO
RICO".
01 HEADING-2.
   05        PIC X(8).

   05         PIC X(18) VALUE
   "RECINTO DE BAYAMON".
01 HEADING-3.
   05        PIC X(7).

   05         PIC X(19) VALUE
   "REGISTRO DE CHEQUES".
01 HEADING-4.
   05 FECHA   PIC X(6)   VALUE

   "FECHA:".

   05         PIC X.
   05 MONTH   PIC 99.
   05        PIC  X VALUE "/".
   05 DAY     PIC 99.
   05       PIC  X  VALUE "/".
   05 YEAR    PIC 99.
01 HEADING-5.
   05        PIC X(7)   VALUE

   "ACCOUNT".

   05         PIC XX.
   05        PIC X(6)   VALUE

   "CHECKS".

   05         PIC XX.
   05        PIC X(5)   VALUE

   "CHECK".

   05         PIC X(11).
   05        PIC X(6)   VALUE

   "CHECKS".

01 HEADING-6.
   05         PIC X.
   05        PIC X(6)   VALUE
```

```
   "NUMBER".

    05          PIC XX.
   05         PIC X(6)   VALUE

   "BEFORE".

    05          PIC XX.
   05         PIC X(6)   VALUE

   "NUMBER".

    05          PIC X.
   05         PIC X(6)   VALUE

   "CHK".

    05          PIC X.
   05         PIC XXX   VALUE

   "AMT".

    05          PIC XXX.
   05         PIC X(5)   VALUE

   "AFTER".

01 DETAIL-1.
    05 FIELD-1 PIC (5).
   05         PIC XXXX.

    05 FIELD-2 PIC 99.
    05         PIC X(5).
    05 FIELD-3 PIC ZZZZ9.
    05         PIC XX.
    05 FIELD-4 PIC Z,ZZZ.99.
    05         PIC XXX.
    05 FIELD-5 PIC 99.
01 TOTAL-1.
    05         PIC X(22).
    05         PIC ZZZ,ZZZ.99.
```

Tome en consideración lo siguiente:

1. El primer caracter del archivo determina el tipo de linea a definir. (H = Heading, D = Detail, T = Total).
2. Líneas en blanco no tienen que definirse.
3. Los ***string*** de los ***values*** van en una segunda línea independientemente de su largo excluyendo el formato de fecha. (los "/")
4. El código tiene que generarse con la menor cantidad de caracteres posibles.
5. Los nombres de variables se incrementan automáticamente.

6. Los signos > y < indican final y principio de un campo. Cuentan como espacio.

Fecha: 04/15/2000	**Nombre de la competencia**: _____
Categoría: Experto	**Universidad**: _____
Autor: Nelliud Torres	**Tipo de competencia**: _____
Problema #: _____	**Algoritmos**: _____

COBOL DESCRIPTION OPTIMIZER

Source File Name: **COBOPT.XXX**

Input File Name: **DESC-I.LIB**

Output File Name: **DESC-O.LIB**

En un centro de cómputos se encontraron con problemas de espacio en disco en su Mainframe principal. Después de utilizar varias técnicas para liberar espacio en disco, se encontraron con el problema de que todavía necesitaban espacio adicional. El *System Manager* sugirió que se depuraran las librerías de las definiciones de archivos de COBOL. Esto debido a que el 80% de su código es en ese lenguaje. Desarrolle un programa que lea de entrada una descripción de record (nivel 01) y le disminuya la mayor cantidad de caracteres posibles. Ejemplo:

DESCRIPCION ORIGINAL	DESCRIPCION OPTIMIZADA

`01 RECORD-DEF.`
`05 CAMPO-1 PICTURE XXXXX.`
`05 CAMPO-2 PICTURE 99999.`
`05 CAMPO-3 PIC 99.`
`05 CAMPO-4 PICTURE 99999V99.`
`05 CAMPO-5 PIC XXXX.`
`05 FILLER PICT X(10).`
`05 CAMPO-6 PIC V9999999.`
`05 FILLER PICTU XXX.`
	`01 RECORD-DEF.`
	`05 CAMPO-1 PIC X(5).`
	`05 CAMPO-2 PIC 9(5).`
	`05 CAMPO-3 PIC 99.`
	`05 CAMPO-4 PIC 9(5)V99.`
	`05 CAMPO-5 PIC X(4).`
	`05 PIC X(10).`
	`05 CAMPO-6 PIC V9(7).`
	`05 PIC XXX.`

Tome en consideración lo siguiente:

7. Los FILLER se eliminan del código original.
8. PICTURE se abrevia a PIC.
9. Cualquier símbolo de editaje mayor de 3 posiciones (X ó 9) se abrevia con la cantidad de caracteres puestos en un número entre comillas.
10. Se tiene que considerar el símbolo V.
11. Se debe mantener la alineación que traiga el cógido original.
12. Asuma que el código no tiene errores de sintaxis.
13. Solo habrá una descripción de record por archivo.

Fecha: 04/15/00	**Nombre de la competencia**: _____
Categoría: Experto	**Universidad**: _____
Autor: Juan M Solá Sloan	**Tipo de competencia**: _____
Problema #: _____	**Algoritmos**: _____

TCAL 5.501 CompilerTCAL Nested Loop to 80x86 Assembly Language

A usted se le ha encomendado crear parte del compilador de TCAL5.501. TCAL es un lenguaje derivado del COBOL que se utiliza para programar terminales de mano de almacén. La tarea que se le ha encomendado es convertir una serie de ciclos con contadores en *Assembly Language* de 80x87.

Los ciclos en Assembly funcionan de la siguiente manera: El registro CX va a tener la cantidad de veces que se ejecutará el ciclo. Cada vez que la instrucción de LOOP aparezca en el código, CX se decrementará automáticamente. Un ciclo de 1 a 10 se convierte en un ciclo de 10 a 1. Ejemplo:

Ciclos en Assembly

```
MOV CX,10
  - Otras instrucciones van aqui -
LOOP
```

Los ciclos anidados dan problema pues el único registro de propósito general que funciona con LOOP es CX. Para hacer que un ciclo anidado funcione se debe empujar el valor de CX en el Stack (Véase el ejemplo de abajo).

Instrucciones de Assembly a utilizarse
Para este programa las instrucciones que se utilizarán serán suma, resta, incremento y decremento. En assembly de 80x86 son:

```
INC X        incremento X
DEC X        decremento X
SUB X,Y      resta X - Y
ADD Y,Z      suma Y + Z
```

Instruccion de LOOP: (Véase ejemplo de ciclo en Assembly)

Ejemplo del insumo (TCAL.IN)	**Salida (TCAL.OUT)**

DATA DIVISION.
01 X PIC 9.
01 Y PIC 9.
01 Z PIC 9.
PROCEDURE DIVISION.
PERFORM VARYING CTR FROM 1 BY 1 UNTIL CTR > 10
PERFORM VARYING CTR2 FROM 1 BY 1 UNTIL CTR > 100
COMPUTE X = X + 1
END-PERFORM
COMPUTE Y = Y-1
COMPUTE Z=Z-X
END-PERFORM.
PERFORM 200-OPEN_PORT
STOP RUN.
	MOV CX, 10
	PUSH CX
	MOV CX, 100
	INC X
	LOOP
	POP CX
	DEC Y
	SUB Z,Y
	LOOP
	CALL OPEN_PORT
	.EXIT 0
	END

Nota: Las instrucciones de PERFORM 200-OPEN_PORT y STOP RUN. Tiene usted que colocarlas al final de su código en Assembly siempre que aparezcan en el archivo de entrada (Substituir por CALL OPEN_PORT y .EXIT 0, END como en el ejemplo). ***No aparezeran otros PERFORM a otras rutinas.***

UNIVERSIDAD DE PUERTO RICO
ADMINISTRACION DE COLEGIOS REGIONALES
COLEGIO UNIVERSITARIO TECNOLOGICO DE BAYAMON
DEPARTAMENTO DE COMPUTADORAS
COMPETENCIAS DE ELIMINATORIA
CATEGORIA PRINCIPIANTE

PROBLEM # 1. CAPS

INPUT FILE NAME: **BEGDIV1.DAT**
OUTPUT FILE NAME **BEGDIV1.OUT**

PROBLEM:

Utilize el archivo **ASCII BEGDIV1.DAT** que esta incluido en el disco que se entrego y el cual contiene la siguiente información:

Universidad de Puerto Rico
Administracion de Colegios Regionales
Colegio Universitario Tecnologico de Bayamon

No se utilizan los acentos para este problema. Haga un programa que cambie las letras minúsculas por mayúsculas y viceversa y lo guarde en el archivo **BEGDIV1.OUT**. Se puede utilizar cualquier función pre-definida que contenga el lenguaje para convertir las letras.

PROBLEM # 2. CHARACTER TO ASCCII TO CHARACTER AGAIN

INPUT FILE NAME: **BEGDIV1.DAT**
OUTPUT FILE NAME: **BEGDIV2.OUT**
OUTPUT FILE NAME: **BEGDIV2A.OUT**

PROBLEM:

Utilizando el archivo de entrada del problema anterior, haga un programa que cree un archivo (**BEGDIV2.OUT**), el cual convierta en ceros(**0**) y unos(**1**) los caracteres, utilizando el código **ASCII** (incluído en estas hojas) del archivo **BEGDIV1.DAT**. En otras palabras se sustituirá cada letra por ocho caracteres simulando el código binario **ASCCII**. Una vez el archivo (**BEGDIV2.OUT**) este creado, el programa procederá a leerlo como archivo de entrada (cerrarlo primero) y lo volverá a convertir en un archivo de caracteres **ASCII** llamado **BEGDIV2A.OUT.**

PROBLEM # 3. PHONE CODE

INPUT FILE NAME: **NONE**
OUTPUT FILE NAME **NONE**

PROBLEM:

Los números telefónicos tienen una serie de letras asignadas las cuales son:

1 = No tiene letras asignadas

2 = **A B C**

3 = **D E F**

4 = **G H I**

5 = **J K L**

6 = **M N O**

7 = **P R S**

8 = **T U V**

9 = **W X Y**

0 = No tiene letras asignadas

Cree un programa que pida de insumo un número telefónico de siete digítos incluyendo el guión pero sin área code. El programa mostrará en la pantalla el número convirtiendo las letras que tenga en números. El programa tiene que validar el dato de entrada:

1. Que existan **8** caracteres(incluyendo guión).
2. Que el guión exista y esté en la posición correcta.
3. Que no exista ningún otro caracter especial aparte del guión, letras y números.
4. No se utilizan las letras **Q** y **Z** por lo que si son incluidas, el programa debe indicar que esas letras no son válidas.

SAMPLE INPUT / OUTPUT:

INPUT: 787-CASH **OUTPUT:** 787-2274

INPUT: THE-BEST **OUTPUT:** 843-2378

PROBLEM # 4. DAY OF THE WEEK

INPUT FILE NAME: **NONE**
OUTPUT FILE NAME: **NONE**

PROBLEM:

Haga un programa que calcule el día de una fecha en particular que por lo menos esté en el siglo 20. El programa recibirá de entrada una fecha con el formato **MM/DD/YY**. Hay que validar:

1. Que la fecha este correcta.
2. Que no sea menor de **1901** ni mayor de **1999**.
3. Los días y los meses. (Ej. No existe un 30 de febrero)
4. Si puede validar un año bisiesto, esto se considerará al momento de evaluar la cantidad de problemas que resolvió el estudiante.

SAMPLE INPUT / OUTPUT:

INPUT: 03/01/96
OUTPUT: VIERNES

INPUT: 06/31/96
OUTPUT: INCORRECT DATE

PROBLEM # 5. TEXT INVERTER

INPUT FILE NAME: **NONE**
OUTPUT FILE NAME **NONE**

PROBLEM:

Haga un programa que acepte de entrada un "string" de caracteres y los invierta. **Para resolver este problema no se puede utilizar ninguna función existente en el lenguaje que haga este proceso.** El programa **tiene** que manejar el "string" por caracteres y no podrá aceptar más de 20 caracteres consecutivos. No se va a utilizar el espacio en blanco como un caracter entre los que se van a invertir. Hay que validar:

1. Que el "string" no sea mayor de **20** ni menor de **2**.
2. Que los caracteres a utilizarse sean de la **A-Z** min. y may. y los números del **0** al **9**.

SAMPLE INPUT / OUTPUT:

INPUT: ARROZ
OUTPUT: ZORRA

INPUT: ROMA
OUTPUT: AMOR

Fecha: ____/____/____	**Nombre de la competencia**:
Categoría: PRINCIPIANTES	**Universidad**: UPR PROGRAMMING
Autor: _____	**Tipo de competencia**: eliminatoria
Problema #: 1	**Algoritmos**:

TITULO DEL PROBLEMA

Source File Name: XXXXXXXX.XXX
Input File Name: XXXXXXXX.XXX
Output File Name: XXXXXXXX.XXX

PROBLEMAS PARA ELIMINATORIA
CATEGORIA: PRINCIPIANTES

PROBLEMA #1

Escriba un programa que lea del usuario cierta cantidad de medidas en galones, una a la vez, y convierta cada valor a litros antes de pedir el próximo valor. La cantidad de conversiones a realizar será entrada por el usuario. La conversión requerida es: **litros=3.785*Galones**. Limite su respuesta a dos (2) lugares decimales. Al terminar la corrida, debe verificar si el usuario desea repetir el proceso;

EJEMPLO DE LA CORRIDA:

Indique el número de conversiones a realizar: 3

Indique el valor # 1 a calcular: 10
10 galones equivalen a 37.85 litros

Indique el valor # 2 a calcular: 13
13 galones equivalen a 49.21 litros

Indique el valor # 3 a calcular: 7
7 galones equivalen a 26.49 litros

Fecha: ____/____/____	**Nombre de la competencia**:
Categoría: PRINCIPIANTES	**Universidad**: UPR
Autor: _____	**Tipo de competencia**: eliminatoria
Problema #: __2	**Algoritmos**:

PROBLEMA #2

Escriba un programa que lea un archivo de texto (codigo.in), y convierta su contenido al código que aparece a continuación: A - '&', B - '#', C = '!', D – '@', E = ']', F = '~', G = '^', H – '%', I = '<', J – '$', K – '?', L = '>', M – '*', N = '/', O = '\', P = '[', Q = '{', R = '}', S = '(', T = ')', U = '+', V – '_', W = '-', X = '|', Y = ':', Z = '='

La conversión del texto a éste código, debe aparecer en otro archivo (codigo.cod). Luego, debe convertir el contenido del archivo en código (codigo.cod) a texto y escribirlo en otro archivo (codigo.out). El texto de los archivos en texto, aparecerá en letras mayúsculas únicamente. El formato del texto debe ser el mismo en el archivo de entrada (codigo.in) y en el archivo de salida (codigo.out).

Fecha: ____/____/____	**Nombre de la competencia**:
Categoría: _principiantes	**Universidad**: UPR
Autor: _____	**Tipo de competencia**: eliminatorias
Problema #: 3	**Algoritmos**:

PROBLEMA # 3

Escriba un programa que lea el usuario 10 valores enteros, y los guarde en un arreglo de una dimensión. Luego el programa debe indicar cual de esos valores es el valor máximo, y cual es el valor mínimo.

EJEMPLO DE CORRIDA

Escriba 10 valores enteros separados por un espacio:
23 12 15 7 54 83 2 37 39 90

El valor máximo de ese grupo es: 90
El valor mínimo de ese grupo es: 2

Fecha: ____/____/____	**Nombre de la competencia**:
Categoría: principiantes	**Universidad**: UPR
Autor: _____	**Tipo de competencia**: eliminatorias
Problema #: 4	**Algoritmos**:

PROBLEMA #4

Escriba un programa que calcule e imprima en pantalla la cantidad de dinero disponible en una cuenta de banco en la cual inicialmente se depositó una cantidad dada de dinero (la cual será entrada por el usuario), y que tiene una tasa de interés del % 8. El programa debe poder calcular este valor dada una cantidad de años (la cual será entrada por el usuario). Use la relación de que el dinero en la cuenta al final de un año, es igual al dinero en la cuenta al principio del año más .08 veces el dinero en la cuenta.

EJEMPLO DE CORRIDA:

Entre la cantidad de dinero depositado en la cuenta: 1000.00
Entre la cantidad de años que desea calcular: 3

Año1: $ 1080.00
Año 2: $ 1166.40
Año 3: $ 1259.71

Proceso Terminado

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C.
(Asociación de Ciencias de Computadoras)

20 de marzo de 1998

FOGUEO DE PROGRAMACIÓN

Problem 1:

(Turtle Graphics) The Logo language, which is particulary popular among personal computer users, made the concept of turtle graphics famous. Imagine a mechanical turtle that walks around the room under the control of a C program. The turtle holds a pen in one of two positions, up or down. While the pen is down, the turtle traces out shapes as it moves: while the pen is up, the turtle moves about freely without writing anything. In this problem you will simulate the operation of the turtle and create a computerized sketchpad as well.

Use a 5-by-50 array floor which is initialized to zeros. Read commands from an array that contains them. Keep track of the current position of the turtle at all times and whether the pen is currently up or down. Assume that the turtle always starts at position 0,0 of the floor with its pen up. The set of turtle commands your program must process are as follows:

Command	**Meaning**
1	Pen up
2	Pen down
3	Turn right
4	Turn left
5	Move forward 10 spaces (or a number other than10)
6	Print the 50-by-50 array
9	End of data (sentinel)

Suppose that the turtle is somewhere near the center of the floor. The following "program" would draw and print a 12-by12-square:

```
2
5, 12
3
5, 12
3
5, 12
3
5, 12
1
```

6
9

As the turtle moves with the pen with down, set the appropriate elements of array floor to 1s. When the 6 command (print) is given, wherever there is a 1 in the array, display an asterisk, or some other character you choose. Wherever there is a zero display a blank. Write a C program to implement the turtle graphics capabilities discussed here. Write several turtle graphics programs to draw interesting shapes. Add other commands to increase the power of your turtle graphics language.

EXPERT DIVISION

DIRECTORY LISTING COMMAND SIMULATOR

Problem 2:

Develop a program that simulate a generic **DIR** command. The input will be read from an input file named **DIR.TXT**. The characters allowed in that file will be:

1. A dot (.) to separate the file name and the extension (optional).
2. Characters from A to Z (Upper and Lowercase will be allowed).
3. Numbers from 0 to 9.

The prompt command will allow the following characters:

1. A dot (.) (optional)
2. An asterisk (*) to substitute one or more characters.
3. A question mark (?) to substitute only one character.

The following are examples of valid commands.

DIR*A.EXE, DIR AB*.COM, DIR A?B?C?.DAT, DIR ABC?H*.*, DIR HELP, DIR*.*, DIR ????????.???, DIR*A?B.*

The rules for filenames will be the same used for MS-DOS. At the end the program will display the totals of files displayed on screen and the total of files read on the file. Remember, the program must be case sensitive.

Sample Output:

COMMAND>DIR*.EXE
ABC.EXE
FINISH.EXE
PROGRAM.EXE
TEST.EXE
Total files read (25). Total files selected (4).

COMMAND>DIR*H.*
FINISH.EXE
ASH.TXT
Total files read (25). Total files selected (2).

COMMAND>DIR A?B?C*.*
Not files found
Total files read (25). Total files selected (0).

COMMAND>DIR A?C*.*
ABC.EXE
AXCTOT.BAT
Total files read (25). Total files selected (2).

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C.
(Asociación de Ciencias de Computadoras)

20 de marzo de 1998

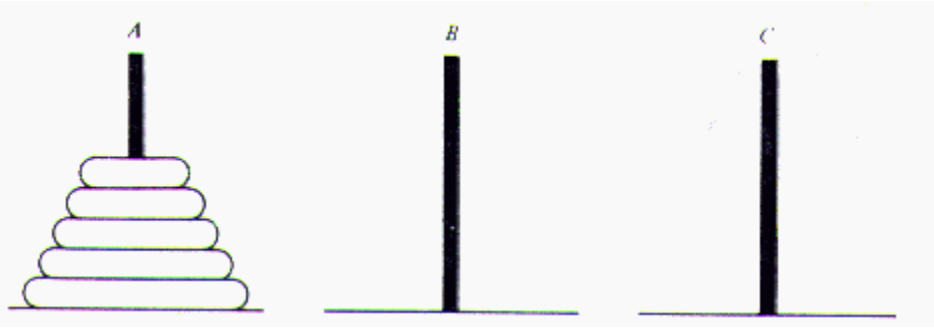
FOGUEO DE PROGRAMACIÓN

Problem 3:

(Towers of Hanoi) Every budding computer scientist must grapple with certain classic problems, and the Towers of Hanoi (see Fig. 5.18) is one of the most famous of these. Legend has it that in a temple in the Far East, priests are attempting to move a peg and arranged from bottom to top by decreasing size. The priest are attempting to move the stack from this peg to a second peg under the constraints that exactly one disk is moved at a time, and at no time may a larger disk be placed above a smaller disk. A third peg is available for temporarily holding the disks. Supposedly the world will end when the priests complete their task, so there is little incentive for us to facilitate their efforts.

Let us assume that the priests are attempting to move the disks form peg 1 to peg 3. We wish to develop and algorithm that will print the precise sequence of disk-to-disk peg transfers.

If we were to approach this problem with conventional methods, we would rapidly find ourselves hopelessly knotted up in managing the disks. Instead, if we attack the problem with recursion in mind, it immediately becomes tractable. Moving disks can be viewed in terms of moving only n-1 disks (and hence the recursion) as follow:



FOGUEO CUTB
Cortesía de: Nelliud D. Torres
Marzo 20, 1998

EXPERT DIVISION

TECO EDITOR :

Problema:

TECO rue un editor de texto famoso que se usó mayormente en la computadora PDP-10 de la compañía DIGITAL. E1 motivo principal de utilizar un editor que trabaje por líncas era debldo a los teletipos que imprimían en papel. Estos teletipos funcionaban igual que un terminal, incluso utilizaban un teclado prácticamente igual a de las computadoras de hoy en día. Sin embargo tenían la desventaja de que no podían manejar pantalla, ya que no se puede controlar esas funciones en papel. De esa necesidad surge TECO el cual permite que se pueda editar un archivo de texto sin la capacidad de manejo de pantalla.

Los comandos de TECO son a base de caracteres y su delimitador es el signo de dó1ar ($). Un signo de dólar significa una separación entre un comando y otro; dos signos de dó1ar ($$) le indica al editor que tiene que ejecutar los comandos dados por el usuario. A continuación una exphcación de los comandos más utilizados: ,!

/

1. J - E1 comando "J" le indica a TECO movimiento en el texto. Las dos formas de utilizarlo para las competencias son las siguientes:
   a. BJ - Mueve' el pointer al principio del buffer.
   b. ZJ - Mueve el pointer al final del buffer.

2. L - Mueve el poiter al principio de la línea que el usuario indique. Por ejemplo:
   a. 0L - Se mueve al principio de la línea en que se encuentre el pointer.
   b. 5L - Se mueve cinco líneas hacia adelante y pone el pointer al principio de la quinta linea.
   c. -1L - Mueve el pointer una línea hacia atrás y coloca el pointer al principio de la línea. Es 1o mismo que L.

3. C - Mueve el pointer "n" cantidad de caracteres. Por ejemplo:
   a. C - Mueve el pointer un caracter a la derecha. Es lo mismo que 1C.
   b 3C - Mueve el pointer trqs caracteres a la derecha.
   c.-1C - Mueve el pointer un caracter a la izquierda.

En caso de que se intente mover el pointer a la izquierda y ya este localizado a la izquierda de la línea, no se enviará mensaje de error y se deja el pointer en esa posición

4. T - Despliega la linea en donde este localizado el pointer o a partir de esa localización.
   Ejemplos:
   a. T Despliega la línea en donde esta el pointer.
   b. OT - Despliega el texto corrido desde el principio del buffer hasta donde se encuentre el pointer.

c. 5T - Muestra las próximas cinco líneas comenzando en la línea en donde este el pointer.
d. HT - Muestra en pantalla todas la líneas del buffer.

5. I - Inserta texto entre las líneas a partir de la localización del pointer. Ejemplo:

Itexto a inclur$ - Inserta el string "texto a incluir".

6. K - Se utiliza para eliminar líneas o texto a partir del pointer. Ejemplos:

a. K - Elimina texto a partir de la localización del pointer hasta el final de la línea.
b. Si el pointer estaba al principio, se dejará la línea en blanco.
c. OK - Elimina texto desde el principio de la línea hasta el pointer.

5K - Elimina las próximas 5 líneas a partir de donde se encuentre el cursor.
Recuerde que si el cursor esta al principio de la línea, esa primera línea debe quedar en blanco.

7. D - Se utiliza para eliminar caracteres a partir de la localización del pointer. Ejemplos:

a. D - Elimina el primer caracter que este a la derecha del pointer
b. 5D- Elimina los pr6ximos 5 caracteres a la derecha del pointer.
c. -2D - Elimina los dos caracteres que estén a la izquierda del pointer. Si el pointer está al principio de la línea, no se elimina nada y no se envía mensaje de error.

"~'

8. EX- Guarde el texto y sale de editor~ El formato es: EX$$

Para comenzar a correr la aplicacón se invocará el siguiente comando: TECO texfile.txt. El archivo ya debe existir ya que para simplificar el problema, el editor no tiene que erear un archivo de la nada, Para representar el pointer, se utilizará el asterisco. Recuerde que hasta que no se incluya el comando "T", TECO no despliega las líneas. Cualquier comando que se salga del rango, el editor lo ignora y no lo ejecuta. A Continuación un ejemplo de como debe trabajar el editor: (Las áreas subrayadas son el output del editor el el resto son comandos de usuario)

```
e:>TECO programa.txt
*0LT$$
*IDENTIFICATION DIVISION
*HT$$
*IDENTIFICATION DIVISION
_____PROGRAM ID. PRUEBA
_____ENVIROMENT DIVISION
____STOP RUN
____*2LT$$
*ENVIRONMENT DIVISION.
____*5C$3D$0LT$$
ENVIR*ENT DIVISION.
_____IONM$0LT$$
ENVIR*ONMENT DIVISION.
_____*EX$$
C.
```

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C

(Asociación de Ciencias de Computadoras)

20 de marzo de 1998

FOGUEO DE PROGRAMACIÓN

Intermediate

Problem 1 · Hotel Reservation

Write an interactive program to manipulate a database. This database manages the hotel reservations. The hotel has 6 rooms, from room 101 to room 106. Room numbers ending in odd numbers are single rooms. Rooms ending in even numbers are double. The program must read: Name, Number of guests, Date of Entry.

Display a menu with the following Options:

1. Reservation \ Check-In
2. Room Status
3. Checkout
4. Statistics
5. Exit Program

Option 1: Access to make a Reservation or Check-In
Option 2: Displays the room status (Occupied, Vacant)
Option 3: Reads the date a guest leaves.
Option 4: Indicates number of times each room has been occupied Option 5: duh!

The program should validate for the correct number of people in each room an that the room isn't occupied. Also validate the exit date-

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C

(Asociación de Ciencias de Computadoras)

20 de marzo de 1998

FOGUEO DE PROGRAMACIÓN

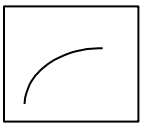
Intermediate

Problem 2: Plot Functions

Write a program that plots polynomial functions.

Input:
order, coefficients and constant

Output: (Graph)



Range of:
x = 1 to 70
y = 1 to 20

Example: (Input)
Order: 3
Coefficients:
Coefl: 4
Coef2:-2
Coef3:2
Constant: 4

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C

(Asociación de Ciencias de Computadoras)

20 de marzo de 1998

FOGUEO DE PROGRAMACIÓN

Intermediate

Problem 3 · Factorials

Write a program that finds the factorial number to any number within I and 25. Validate the input for the range. To exit the program enter X.

Example 1:

Input:

Enter the number: 3

Output:

Factorial of 3! = 6

Example 2:

Input:

Enter the number: 25

Output:

Factorial of 25! = 15511210043300000000000000

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C

(Asociación de Ciencias de Computadoras)

20 de marzo de 1998

FOGUEO DE PROGRAMACIÓN

Intermediate

Problem 4: Equal to Zero

Read a sequence of digits from I to N (where N <= 9) in increasing order:

I 2 3 4 5...N

Insert either a + (for addition) or a - (for substraction) between each of the digits so that the resultant sum is zero.

Diaplay all possible combinations that sums zero.

Example:

Input:

Number = 7

Output:

1+2-3+4-5-6+7=0

1+2-3-4+5+6-7=0

1-2+3+4-5+6-7=0

1-2-3-4-5+6+7=0

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

(Asociación de Ciencias de Computadoras)

20 de marzo de 1998

FOGUEO DE PROGRAMACIÓN

Beginners

Problem 1: Distance, Midpoint and Slope

Write a program that reads two points, (XI, Y1) and (X2, Y2), from a user and find:

a) the distance between the points

b) the midpoint of the line segment

c) the slope of the line trace from one point to another
(If not defined indicate so):

d) the ecuation in the form point-slope

Distance Formula

d(P1, P2)= ((X1-X2) 2+ (Y1-Y2) (1/2)

Midpoint Formula:

(X1+X2)/2, (Yl+Y2)/2

Slope:

m= (Y2-Yl) / (X2-XI)

Display the distances, the midpoint, the slope and the ecuation.

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

(Asociación de Ciencias de Computadoras)

A.C.C

20 de marzo de 1998

FOGUEO DE PROGRAMACIÓN

Beginners

Problem 2: Draw Shapes

Write a program that draws different shapes at random.

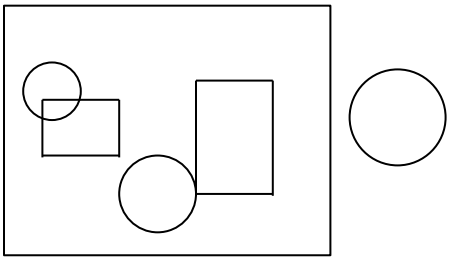
Example:

Input:

Enter number of shapes: 5

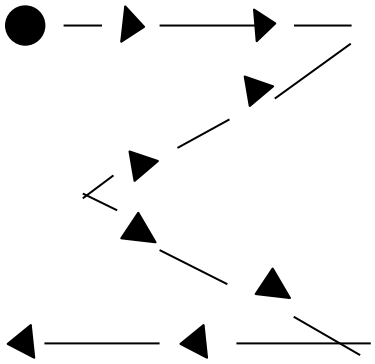
Output:

Screen



PROBLEMA #5

Escribe un programa que utilizando funciones de gráficos dibuje una pelotita, la cual seguirá la moción indicada en el próximo dibujo.



Nota: la pelotita no puede salir del margen de la pantalla, y la misma debe estar lo más cercano al borde posible.

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C

(Asociación de Ciencias de Cmputadoras)

20 de marzo de 1998

FOGUEO DE PROGRAMACIÓN

Beginners

Problem 3: File Management

Write a program that reads from one file and writes to another file.

Read from the inventory file (Invent. in) and read all the data of the items that begin with 10 and store in another file (Invent. out). Give indications to the user that the program, is Reading, Writing, Thinking and Done. At the end display how many items were moved.

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C

(Asociación de Ciencias de Cmputadoras)

20 de marzo de 1998

FOGUEO DE PROGRAMACIÓN

Beginners

Problem 4: Ardes Labels

Write a program that reads an address and displays on the screen in proper order.

Example:

Input: John H. Doe, Hc-02 Box 3769, Levittown, P.R, 00943

Output:

Doe, J.H.

Hc-01 Box 3769

Levittown, PR 00943

The name used will consist of a first name, middle initial and last name (In that order) and then prints the last name, followed by a comma and the first AND middle initial, each followed by a period.

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C

(Asociación de Ciencias de Computadoras)

21 de febrero de 1998

Competencias: Estudiantes de Nuevo lngreso

Ejercicio #1:

Una compañía desea transmitir data a través de las líneas telefónicas, pero su preocupación es que la información sea interceptada indebidamente por otras entidades. Toda su data es transmitida en forma de enteros de cuatro (4) dígitos. Dicha compañía ha solicitado sus servicios para crear un programa que codifique su data para que su transmisión sea más segura.

Su programa debe leer un entero de cuatro(4) dígitos y codificarlo utilizando la siguiente formula: cambie cada digito a: **(La suma de ese digito + 7) mod 10**. Luego intercambie el primer dígito por el tercero y el segundo digito por el cuarto.

Presente el resultado en pantalla.

Universidad de Puerto Rico
Administración de Colegios Regionales
Colegio Universitario Tecnológico de Bayamón

A.C.C

(Asociación de Ciencias de Cmputadoras)

21 de febrero de 1998

Competencias: Estudiantes de Nuevo lngreso

Ejercicio #2:

Decodifica el código generado por el primer programa (Ejercicio I).

Su programa debe leer este código entero de cuatro dígitos, descodificarlo y presentado en pantalla.

Competencias: Estudiantes de Nuevo Ingreso

Ejercicio #3:

El propósito del programa es crear un algoritmo que lea un mensaje del archivo **MENSAJE.TXT.** Este mensaje debe ser traducido a código Morse· En pantalla debe aparecer el mensaje tal y como está en el archivo y luego debe presentar su traducción.

Código Morse: '

a= .-	j= .---	r= .-.
b= -…	k= -.-	s= …
c= -.-.	l= .-..	t= -
d= -..	m= --	u= ..-
e= .	n= -.	v= …-
f= ..-.	ñ= --.--	w= .--
g= --.	o= ---	x= -..-
h= ….	P= .--.	y= -.--
i= ..	q= --.-	z= --..

1= .----	6= -….
2= ..---	7= --…
3= …--	8= ---..
4= ….-	9= ----.
5= …..	0= -----

Ejemplo de corrida:

Mensaje : Hola.

Salida:

Hola.

….

.-..
.-

Categoría intermedio

PROGRAMA DECODIFICACIÓN DOBLE

E1 programa anteriormente mencionado consiste en una doble decodificación de datos Estructurada en la siguien manera la letra A debe ser reeplasada por el espacio en blanco la letra B debe ser suplantada por la letra Z y mismo consecuentemente Por todo el abecedario. Despues de haber concluido con la inversión la salida del progran debe dar su valor numerico en respecto a la posición de el dato dentro del abecedario (el espacio en blanco es despues de la letra Z.

Ejemplo:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	-
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

E1 (-) = Espacio en blanco

Entrada= t o m a
Codifucación nivel 1 =
Codifucación nivel 2 = 14 17 21 27

La decodificación del mismo debe ser entrada en números y el resultado debe ser la entrada original en el ejercicio anteriormente mencionado el mismo debe ir a el archivo encry.dat

Intermedio

Crear un programa que simule la registración de entrada y salida de un hospital. E1 Hospital solo acepta 10 pacientes. E1 proceso de registración debe de estar siempre online hasta que el comando shutdown sea seleccionado en el menú. Los pacientes de alta se sacan de la lista activa y se colocan en una lista pasiva. E1 programa debe aceptar el nombre del paciente, el número de seguro social, dia y hora que entró y el cuarto donde se va a colocar. En el hospital hay solo 5 habitaciones de la habitación 101 a la 105. El programa debe tener un menú para manejar la registración. E1 menú debe tener las opciones de aceptar un paciente, dar de alta, cambiar a un paciente de cuarto y reportes. En reportes debe haber otro menú con las opciones de imprimir la información en pantalla de un paciente buscada por seguro social, la opción de cuartos para ver la capacidad disponible y la opción de pacientes dados de alta.

Fecha: ____/____/____	**Nombre de la competencia**: _____
Categoría: Beginners	**Universidad: CUB**_____
Autor: _____	**Tipo de competencia**: Eliminatoria
Problema #: _____	**Algoritmos**: _____

Crear un programa que transforme un string a otro string utilizando la cantidad menor de ediciones. Estas ediciones se hacen con los comandos de Delete para borrar, Right para moverse a la derecha e Insert para colocar datos.

El programa debe indicar cuantas ediciones se llevaran acabo. El programa debe seguir pidiendo un Nuevo string hasta que uno diga lo contrario.

Problem #1:

An HTML Link Generator is a program that receives an Input file ("PROBI.IN") that follows a Set of rules and generates an output file ("PROEI.our'} accordingly. Make an FITML Link Ganerator that creates an output fite following these rules:

['Ink l's namej;[IInk l's addmssl
['Ink 2's namej;[1Ink 2's addmssj

The output file ("PROEl .our'} should be as follows**:**

<Li>
<A HREF="llInk i•S addmss]TM> link i,s name <*IA*>
<Li> ~ FIREF="gink Ts addmss]"> link Ts name <IA>

The progmm should work no matter what the names of the links am, what the addresses of these links are, or how many links them are in the file.

For the Input file, "PROB1.IN”":

Yahoo's HP;http://www.yahoo.com
Test Page;http://mysite.mynode.org

the output file, "PROB1.QuT,~' should be:

<UL.>
<LI> <A HREF=" http://www.yahoo.com~'> Yahoo's HP </A>
<LI> <A HREE=~http://mysite.mynode.org"> Test Page </A>
</UL>

<u>Problem #2</u>:

Make a programm that outputs the computer clock's time, simulating the characters from a digita1 clock.

Exmple:

If the computer's time upon execution is: 9:21 AM, the progmm should output:

9:21 A

If the hour wele "PM," Instead Of an "A," a "P" should be written after the time.

Problem #3:

Make a program that reads an encoded file (npRO63.INI.), decodes It, and writes the decoded data to an output file ('~PROB3.OUT”). The codlicatlon will be given by the following scheme:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
ZYXWVUTSRQPONMLICIIHGFEDCBA

that is, 'A" codifies to -Z" and vice-versa, "U" codifim to non and vIce-versa, and “N” codifies to belt All chamcters that am not letters will be copied unchanged to the output file, but lowercase lettem will be decoded to their corresponding letter (wnlffen In uppercase), so that an "r" would codify to a "J" and a "Z^{TM} would codify to an ~A" In the output file.
In your ImplementatIon, YOU MAY NOT USE ARRAYS FOR DECODING. You have to design an algorithm to cany out the decoding process, but you may NOT use armyn that contain conversion tables. The progmm should wort for ANY text contained In ~PROB3.IN,'Y but the test file that you have been given Is the following:

AbCDEFGhIJKIMNOpQRStUVWXyZIJ

After execution, the output fire, 'PROBS.OUT," should contain the following:

ZYXWVUTSRQPONMLKJIHGFEDCBA1_!

Problem #4:

Make a progmm that asks the user for the elements (Integer numbers) of a square matrix containing 3 rows and 3 columns, and calculates and prInts out the determinant of said matrix. Example of how to calculate the determInant of a 3x3 matrix:

```
[1    2    3]
 [4    5   -6]
[7    8    9]

 Det=1 x [5x9-(8x6)]-2x(4x9-(7x6)+3x[4x8.(7x5)]
                = -72
```

That is, one takes the first number of the first row (In this case, a 1) and one multlplies it by the cross-product offlie four numbers that am NOT contained in the same row and column as the 1. In this case, these four numbem am 5, -6, 8 and 9. Then, one multiplies the first number with the third' (5 x 9), and then SUBSTRACTS the product of the fourth number and the second (8 x -6). Then, one SUBSTRACTS the second number on the first row (in this case, 2) multiplied by the cross-product of the numbers that am not in the same row and column as the 2, and finally, one ADDS the third value on the first row (here, 3) multiplied by the corresponding cross-product

Input

```
Enter the values for Row 1:  8- 2 0
Enter the values for Row 2:     57  6
Enter the values for Row 3:     41  5
```

Output

The determinant of the matrix is. 234

Problem #5:

Make a program that splits the file specified by the user on the command line into smaller files, where the size of these files is also specified on the command line. Example:

C:/ prob5 prob5.in 4

will break the file called “prob5.in” in smaller files of 4 characters each. These files will be called “p000”, “p001", etcetera, that is, the first letter of the file specified followed by three numbers, which will go from 000 to 999 (if necessary). If the file does not exist, an error message should be given.

The program should work for any file that is specified on the command line, but if the test file:

“prob5.in":

Esta es una prueba

is broken In files 6 characters long each, these would be:

"p000":

Esta e

"p001":

s una

"p002":

prueba

programa: prog1.xxx
archivo de entrada: prog1.in
archivo de salida: prog1.out

Problema #1: Resta de Números Grandes

Escriba un programa que reste dos números de hasta 32 cifras de largo.

Asuma:

1. Los dos números se encuentran en una misma línea separados por un espacio.
2. Ambos números son enteros positivos aunque la diferencia no tiene que ser positiva.

data de ejemplo:

400 321
1000 2001

salida:

400 - 321 = 79
1000 – 2001 = -1001

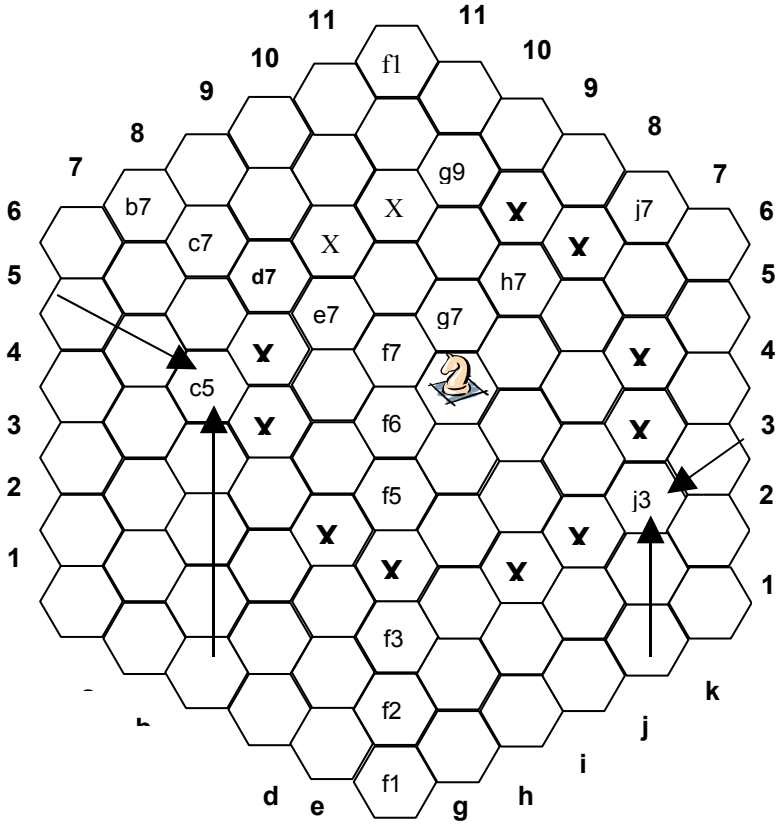
programa: prog2.xxx

archivo de entrada: prog2.in

archivo de salida: prog2.out

Problema #2: Camino del Caballo

En Ajedrez Hexagonal, el paso del caballo se define como muestra el diagrama. Las 12 equis muestran los doce posibles pasos de este desde la casilla g6.



Problema: Escriba un programa que calcule el camino que el caballo debe tomar desde un hexágono inicial a otro.

Asuma:

1. Las posiciones del tablero se escriben ‘xn’ donde ‘x’ es la letra de la columna, y ‘n’ es el numero de línea.

f

2. Cada línea del archivo de

entrada esta en el formato xx-yy donde xx es la posición de arranque del caballo, y la posición final. Todas las posiciones en este archivo se encuentran en el tablero.

Data de ejemplo:
j3-c5

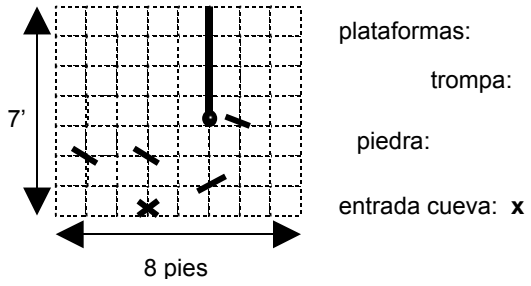
salida:

Un camino desde j3 hasta c5 es j3-j6-h4-f7-c5

programa: prog3.xxx
archivo de entrada: prog3.in
archivo de salida: prog3.out

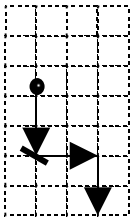
Problema #3: *El elefante Furioso*

Un elefante esta muy furioso porque unas ratas han construido una cueva en el fondo de su pozo de agua fresca. Por lo que esta ha decidido meter la trompa en el agua y tapar la entrada de la cueva con una de las piedras que se encuentran en el área. Desafortunadamente, el elefante se encontró con dos problemas: primero,como el pozo es muy profundo no alcanza la entrada, por lo que tendrá que soltar la piedra, y segundo, que no la puede soltar directamente, porque las ratas, cansadas de tanto bucear, han construido además unas plataformas inclinadas en las que hacen escala al entrar y salir del agua.



Cuando el elefante suelta la piedra, esta acumula velocidad, y rebota cuando choca con una de las plataformas. Verdaderamente, cada vez que la piedra cae, esta acumula velocidad. Por otro lado, la piedra la pierde cuando sube o cuando se mueve para la izquierda o la derecha. Observe el siguiente ejemplo:

La piedra cae 2', choca con la plataforma, se desplaza 2' a la derecha, y cuando pierde toda la velocidad, vuelve a caer.



Las piedras pueden chocar con las plataformas por encima o por debajo. El efecto es siempre el mismo: la piedra puede subir o moverse para los lados la misma cantidad de pies que baja.

Problema: Dado que el elefante puede hundir la trompa en cualquier parte del pozo unos 4 pies, determine de donde debe soltar la piedra para que esta tape la entrada del la cueva.

Asumir:

El pozo mide 7 pies de ancho y 7 pies de profundidad.

El elefante puede mover el ultimo pies de la trompa, por lo que puede, dado que

no se encuentre ninguna plataforma en el camino, hundir la trompa 3 pies y I para la

izquierda o la derecha.

El archivo contiene un mapa del pozo, donde los puntos (.) llenan los espacios de agua,

las plataformas se representan con '\' o '/', dependiendo de la dirección de esta, y la

entrada de la cueva se marca con una x. La entrada de la cueva siempre esta en el fondo.

Si la piedra pierde todo velocidad, y se encuentra posada sobre el piso o una plataforma,

no se mueve de ahí.

Si la piedra choca contra una de las paredes, esta rebota en la dirección contraria.

Se garantiza que el mapa tiene solución.

El girar la trompa se debe especificar como a la izquierda, derecha, o no es necesario.

data de ejemplo:

. . . . . .

. . . . . .

. . . . . .

. . . . . \ .

. . . \ . . . .

. . . . . / . .

. . . x . . . .

salida:

distancia del borde: 5'

profundidad: 4'

girar: no es necesario.

Fecha:	**Nombre de la competencia**:
Categoría:	**Universidad**:
Autor:	**Tipo de competencia**:
Problema #:	**Algoritmos**:

NOTA GENERAL: LA VALIDACION DE DATOS ES REQUESITO EN TODOS LOS PROBLEMAS.

PROBLEMA #1

Escriba un programa que realize una serie de conversiones de Grados Farenheith a Grados Celcius. El mismo debe pedir del usuario el primer valor de la serie, y la escala en la que se desea que aumente la escala. Además, el usuario también entrará el número de veces que desea que se haga la conversión.
Nota: la ecuación de conversión es la siguiente: (Farenheith – 32)/1.8

Ejemplo de Corrida:

Entre el primer valor de la serie: 50
Entre la escala en que desea que aumente la serie: 4
Entre el número de conversiones que desea realizar: 3

Grados Farenheith	Grados Celcius
50	10
54	12.22
58	14.44

PROBLEMA #2

Escriba un programa que leerá del usuario un caracter, y luego, dirá si el caracter es Mayúscula o Minúscula; y lo imprimirá en pantalla, justo en la posición del alfabeto que le corresponde. Los espacios antes, y después del caracter, deben estar oxupados por '-'.
Nota: Se utilizará el alfabeto americano.

Ejemplo de corrida:

Entre un caracter: d

El caracter es minúscula.
---d----------

PROBLEMA #3

Escriba un programa que pida del usuario un nombre de un empleado, y un total de horas trabajadas y dada esta información haga los calculos correspondientes e imprima la siguiente información:

a) Nombre del empleado
b) El salario bruto del empleado
c) La cantidad a descontar por concepto de impuestos
d) La cantidad a ganar por concepto de horas extras
e) La cantidad a descontar por concepto de plan médico
f) total de deducciones
g) Salario neto

Nota: Paga por hora, hasta 40 horas: $6.00

Cada hora por encima de 40 horas: $7.50
Impuesto: 7.5% del salario bruto. Si el salario bruto es mayor de $300, el impuesto será 8.5%
Pago por plan médico: 6% del salario bruto. Si el salario bruto es mayor de $275, el pago por plan médico será de 7.2% del salario bruto.
Salario Neto: Salario Bruto menos deducciones.

Ejemplo de Corrida:

Entre el nombre del empleado: Valmarys Ramírez
Entre el total de horas trabajadas: 36

Nómina de Valmarys Ramírez
Salario Bruto: $216
Deducción por impuesto: $16.20
Deducción por plan médico: $12.96
Total deducciones: $20.96
Salario Neto: $195.04

PROBLEMA #4

Escriba un programa que sume dos números binarios, con un máximo de ocho (8) dígitos por **números.** Recuerde, que en binarios 1+1=0 y se "lleva 1"; 1+0=1; y 1+1+1=1 y se "lleva 1".

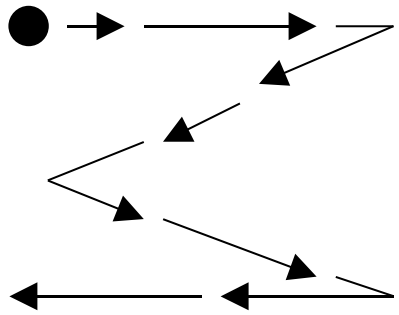
Ejemplo de corrida:

Entre el primer número a sumar: 10011
Entre el segundo número a sumar: 10101

La suma de éstos números es: 101000

PROBLEMA #5

Escribe un programa que utilizando funciones de gráficos dibuje una pelotita, la cual seguirá la moción indicada en el próximo dibujo.



Nota: la pelotita no puede salir del margen de la pantalla, y la misma debe llegar lo más cercano al borde posible.