"First, solve the problem. Then, write the code" John Johnson.

ProgramaMe 2023

Regional online de Ourense

Listado de problemas

A - En busca del radón

B - El Tap Code

C – Remontando

D - Hacer leña del árbol caído

E - Palabras Abecedarias

F – MVP

G - Conducción eficiente

H - Operación camello

Autores de los problemas

- Víctor Manuel Alonso Rodríguez (CIPF A Carballeira – Marcos Valcárcel)
- Juan Nieto González (CIPF A Carballeira – Marcos Valcárcel)
- Juan José Lorenzo Mayo (CIPF A Carballeira – Marcos Valcárcel)
- María Luz Amaro Lois (CIPF A Carballeira – Marcos Valcárcel)
- Julio Mosquera González (CIPF A Carballeira – Marcos Valcárcel)

A - En busca del radón

El radón es un gas radioactivo que se genera de manera natural por la descomposición del uranio presente en las rocas graníticas. Si el gas sale directamente a la atmósfera no supone ningún riesgo, pero si se filtra hacia el interior de los edificios y se producen acumulaciones altas de radón, los habitantes de las viviendas corren el riesgo de padecer cáncer de pulmón. En el subsuelo de Galicia este mineral es muy abundante, por lo que la Universidad de Santiago de Compostela está haciendo un estudio de las zonas donde hay mayor presencia de granito, para intentar relacionarlo con los casos de cáncer de pulmón detectados.

Para hacer el estudio, se dividió la superficie de la comunidad de Galicia en cuadrículas, y se analizó cada cuadrícula para ver si hay presencia de granito. A las cuadrículas que contienen granito se las denomina granitepoint. Si dos granitepoints están contiguos, formarían parten de una veta de granito, la cual puede estar formadas por varios granitepoints. Tenemos que ayudar a los investigadores de este estudio, calculando el número de vetas de granito que hay en una determinada superficie de estudio.

Entrada

En primer lugar, se leerá el número de zonas de terreno que se van a analizar. A continuación, se leerán dos valores enteros x e y, en la misma línea y separados por un espacio en blanco, que representan el número de filas y de columnas en las que se dividió la zona a analizar. Después, se leerán x filas de y caracteres cada una de ellas. Los caracteres pueden ser una ‘g’, que indica que en esa cuadrícula hay granito (es un granitepoint), o un carácter ‘-‘ que indica que no hay granito.

Salida

Para cada terreno analizado, se escribirá el número de vetas de granito que contiene, teniendo en cuenta que dos granitepoints forman parte de la misma veta si están contiguos vertical, horizontal o diagonalmente.

Entrada de ejemplo

```
3
3 5
gg---
----g
g--g-
1 4
-ggg
1 5
-----
```

Salida de ejemplo

```
3
1
0
```

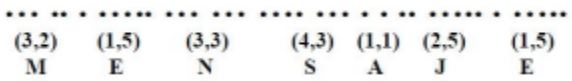
B - El Tap Code

El Tap Code es una forma de encriptar mensajes usada por prisioneros en Vietnam. Es un método muy simple que consiste en traducir cada letra de un mensaje en una serie de sonidos (taps) hechos con metal, madera, etc. y pausas. También puede ser representado gráficamente. En este caso cada letra se traduce en dos series de puntos separados por un espacio, que representan dos números indicando la fila y columna que corresponde a la letra en una matriz de claves de 5x5.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I	J
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

En la matriz de la figura, suponiendo que cada asterisco significa un tap, la letra **D** se representaría: * ****
Los asteristos indican un tap para la primera fila, y 4 indicando la columna (1, 4).

La palabra “MENSAJE” se traduciría de la siguiente manera.



Una consideración importante es que la letra K se codifica como si fuera una C.

Entrada

Comienza con un número entero que indica la cantidad de palabras a codificar. A continuación, las palabras que se codificarán. Estas palabras pueden contener letras en mayúscula o minúscula sin acentuar, y ninguna contendrá la letra Ñ.

Salida

Para cada palabra, se mostrará en una línea la palabra codificada en Tap Code. Para ello se muestra la secuencia de asteriscos que representa cada letra, separando los taps de filas y columnas por un espacio en blanco. Cada letra se separará también con otro espacio en blanco de la siguiente.

Entrada de ejemplo

```
3
MENSAJE
KILO
PROGRAMAME
```

Salida de ejemplo

```
*** ** * ***** *** *** **** *** * * ** ***** * *****
* *** ** **** *** * *** ****
*** ***** **** ** *** **** ** ** **** ** * * *** ** * * *** ** * *****
```

C - Remontando

Los partidos de fútbol son como montañas rusas, donde el resultado final se puede ir transformando con pequeñas remontadas de uno u otro equipo, y no siempre refleja lo sucedido a lo largo del partido, con sinuosos cambios de tercio que hacen que la afición anime eufóricamente.

Una de las aficiones más importantes de Galicia nos ha encargado un programa que nos indique en cuantas ocasiones ha estado el marcador igualado a lo largo de un partido. En un partido con el siguiente escenario debemos concluir que hubo 3 momentos de igualdad.

Minuto	Marcador
0	**0-0**
15	1-0
18	2-0
19	2-1
25	**2-2**
48	2-3
82	**3-3**
94	4-3

Entrada

El programa deberá leer, por cada caso de prueba, un primer número $0 \leq n \leq 100$ que indicará los goles que se anotaron en el partido. Los casos de prueba terminan cuando introducimos un valor negativo de goles.
A continuación, vendrán n líneas detallando el momento en que se anotó cada gol precedido de las letras L (local) o V (visitante) en función de quien ha sido el que ha anotado el gol. El tiempo se expresará en el siguiente formato mm:ss, indicando mm los minutos y ss los segundos. Por razones desconocidas, los tiempos de anotación de cada gol se proporcionan en orden arbitrario.

Salida

Por cada caso de prueba se escribirá el resultado final del partido con los goles anotados por el equipo local y el equipo visitante separados por un guión '-', a continuación se mostrará un espacio en blanco y el número de veces en los que el tanteo ha estado igualado (mismo número de goles para cada equipo).

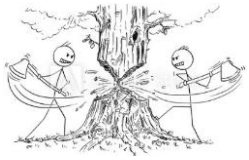
Entrada de ejemplo

```
4
L 8:15
L 18:34
V 3:07
L 67:12
5
L 8:03
V 18:19
V 8:59
L 81:25
L 64:40
-1
```

Salida de ejemplo

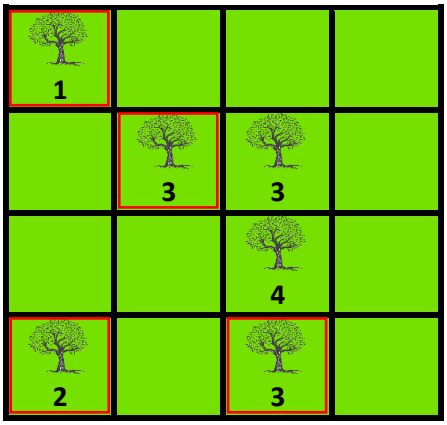
```
3-1 2
3-2 3
```

D - Hacer leña del árbol caído



Los buenos cortadores de árboles intentan hacer el menor daño posible al derribar un árbol, esto es, no derribar por el camino otros que estén cerca (en su radio de caída), o que la caída tenga lugar fuera de los lindes de la finca propietaria del árbol.
Vamos a suponer que las fincas son rectangulares y que cada árbol ocupa una posición dentro de la finca. Se debe almacenar la altura de cada árbol en unidades equivalentes a posiciones/cuadrículas de la finca. De este modo sabremos si podremos o no derribar un árbol haciendo el menor daño posible.

Los árboles sólo pueden ser tirados ocupando las cuadrículas próximas en vertical, horizontal o diagonal, y siempre y cuando no haya ningún árbol en dichas cuadrículas.
Debemos tener en cuenta que si un árbol tiene el valor de altura 1, va a poder ser derribado siempre, ya que solamente ocuparía su cuadrícula en la caída. En el siguiente ejemplo podremos derribar los 4 árboles señalados.



Entrada

El programa deberá procesar múltiples casos de prueba, representando cada uno una finca formada por un rectángulo de pequeñas cuadrículas de igual tamaño en los que puede haber hueco para un árbol de una determinada altura.
Cada caso de prueba comienza con dos números $1 \leq c, f \leq 100$, indicando el número de columnas y de filas de la finca de cuadrículas. A continuación, y en la misma línea, se indica el número a de árboles de la finca.
Si hay árboles, en la siguiente línea aparecen la posiciones de todos ellos, indicando para cada uno la columna (1…c) y la fila (1…f) que ocupan así como su altura h. En total, aparecerán $3 \times a$ números separados entre sí por un espacio en blanco.
La entrada termina con una línea con tres ceros (finca con dimensiones nulas y sin árboles), que no debe procesarse.

Salida

Por cada caso de prueba el programa indicará, en una línea, el número de árboles que pueden derribarse sin dañar a otro directamente, o salirse de los límites de la finca.

Entrada de ejemplo

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

Salida de ejemplo

```
3
5
1
```

E - Palabra Abecedaria

El club de lectura YoLeo debe promover el gusto por la lectura, que los niños desarrollen el hábito lector y enriquezcan su vocabulario. Para lograrlo, la imposición de las lecturas no resulta la mejor forma. Para hacer compatible la evaluación escolar de las lecturas con el "placer por leer", el club decidió estimularlo con juegos.
De esta forma surgió la idea buscar palabras **abecedarias**. Una palabra se dice abecedaria si las letras que la forman se encuentran en orden alfabético, cuando son leídas de izquierda a derecha. Son ejemplos de palabras abecedarias: Amor, filo, chintz, luz, dinos.

Para comprobar si los niños eran capaces de encontrar correctamente las palabras abecedarias, se nos ha pedido elaborar un programa que sea capaz de extraer todas aquellas presentes en un texto.

Entrada

La entrada estará formada por líneas de longitud variable sin signos de puntuación ni caracteres especiales. La longitud máxima del texto que se admite por cada línea es de 1000. Una línea con el texto "ABCD" indica el fin de la entrada.

Salida

Para cada línea de entrada, se obtendrán el número de palabras abecedarias diferentes que se encuentran en el texto. Para considerar que dos palabras son iguales no se hace distinción entre mayúsculas y minúsculas.

Entrada de ejemplo

```
El himno del amor
Dinos si los ajos se colocan en una mesa de madera con la luz del sol de la mañana
Al filo de lo imposible
ABCD
```

Salida de ejemplo

```
4
7
4
```

F - MVP

En las competiciones de baloncesto, una de las decisiones más controvertidas puede ser la elección final del mejor jugador, o MVP, del torneo.

Conscientes de ello, los organizadores de la competición “Gatolandia 2020” han decidido desarrollar un algoritmo que, a partir de las estadísticas de todos los jugadores nos devuelva cuál ganará el trofeo de mejor jugador del partido.

La valoración de cada jugador se calcula sumando los puntos anotados, los rebotes y recuperaciones. A esta cantidad se restan las pérdidas de balón, los tiros realizados y las faltas cometidas.

Entrada

La entrada comenzará con una primera línea que contiene el nombre del primer equipo, seguido de una serie de líneas con los nombres de todos los jugadores junto con sus estadísticas de juego separadas por comas, hasta una línea en la que aparecerá, nuevamente, el nombre del primer equipo, y que limita el fin de los datos de dicho equipo. En la siguiente línea aparecerá el nombre del segundo equipo, seguido nuevamente por todos sus jugadores con sus datos. La entrada terminará con la repetición del nombre del segundo equipo.

La información de cada jugador mostrará, obligatoriamente, los siguientes valores separados por comas y por este orden:

Nombre del jugador, minutos, puntos, rebotes, recuperaciones, pérdidas, tiros realizados, tiros convertidos y faltas.

Salida

Se mostrará una lista de jugadores ordenada por valoración, en la que aparecerá en una línea el nombre de cada jugador que haya disfrutado de minutos en la cancha, seguido de su valoración entre paréntesis.

En caso de igualdad, se seleccionará en primer lugar al jugador que hubiese disputado menos minutos. Si han jugado los mismos minutos se ordenarían por mayor número de tiros convertidos. De persistir la igualdad, se ordenarían por nombre en orden alfabético.

Entrada de ejemplo

```
T.I.A. BASKET CLUB
Mortadelo,20,22,20,30,60,100,15,0
Filemon,20,0,0,0,20,1,1,4
Profesor Bacterio,40,40,0,40,2,20,17,3
Superintendente Vicente,20,0,0,0,20,1,1,5
Ofelia,40,40,0,40,2,20,17,2
Irma,0,0,0,0,0,0,0,0
T.I.A. BASKET CLUB
MARVEL CLUB
Capitan America,20,20,20,30,60,100,15,0
Hulk,1,0,0,0,0,0,0,5
Thor,20,20,20,30,60,100,15,2
Viuda Negra,20,20,20,30,60,101,15,1
Vision,40,40,0,40,2,20,17,2
Soldado de Invierno,0,0,0,0,0,0,0,0
MARVEL CLUB
```

Salida de ejemplo

```
Ofelia (56)
Vision (56)
Profesor Bacterio (55)
Hulk (-5)
Filemon (-25)
Superintendente Vicente (-26)
Mortadelo (-88)
Capitan America (-90)
Thor (-92)
Viuda Negra (-92)
```

G - Conducción eficiente

Hoy en día existe una gran conciencia con respecto a la preservación del medio ambiente y el cambio climático, lo que conlleva que en un futuro muy cercano todos los coches serán eléctricos. Además, en la conciencia de los conductores está el intentar lograr una conducción eficiente para contaminar lo menos posible, gastando la mínima cantidad de energía. Por ello, cada vez más coches nos indican si estamos conduciendo o no de manera eficiente.

El acelerador de un coche envía continuamente información a la centralita acerca de su posición, concretamente el porcentaje de su recorrido. Si envía un 100% significa que el pedal está pisado a fondo, mientras que si envía un 0% no se está acelerando en ese momento. Esta información tiene diferentes propósitos, pero uno de ellos es informar al conductor acerca de su manera de conducir.

Cada vez que varía la posición del acelerador, se envía a la centralita el porcentaje del recorrido total del pedal "pisado", así como el tiempo en milisegundos en que el pedal se mantuvo en una determinada posición. Toda esta información queda registrada, separando todos los valores con un carácter #. La secuencia ## indica que el vehículo finaliza el recorrido.

Por ejemplo, la secuencia: 28#26#85#34#50#60##

Indica que el vehículo inicia la marcha con el acelerador pisado en un 28% de su recorrido, circula así durante 26 milisegundos, momento a partir del cual el pedal se pisa a un 85%. Sigue así durante 34 milisegundos, momento en el que el acelerador cambia a un 50% y se mantiene durante 60 milisegundos. Como a continuación encontramos la secuencia ##, quiere decir que el coche ha finalizado su recorrido.

Ayuda a los ingenieros a programar la centralita para informar al conductor de si fue o no eficiente durante su conducción.

Entrada

Se procesarán múltiples casos de prueba, representado cada uno de ellos con una línea que contiene la información almacenada en la centralita. La información está formada por una secuencia de dos valores numéricos separados por el carácter #. El primero representa la posición que tiene el acelerador, y el segundo el tiempo en milisegundos en que se mantuvo en dicha posición. La aparición de la secuencia ## indica la finalización del recorrido.

La entrada finaliza con una línea con el texto STOP TEST.

Salida

En función de lo eficaz que fuera el conductor se visualizará uno de los mensajes siguientes:

- Si la mayor parte del trayecto ha transcurrido con el acelerador pisado en un porcentaje comprendido entre 0 y 50% (incluido), se trata de un conductor eficiente, y se mostrará uno de los siguientes mensajes:
    - VERY EFFICIENT DRIVING, si ha estado más tiempo entre 0 y 25% (incluido).
    - EFFICIENT DRIVING, si ha estado más tiempo entre más de 25% y 50% (incluido).

- Si la mayor parte del tiempo el acelerador ha estado pisado en un porcentaje comprendido entre más de 50% y 100% (incluido), el conductor es poco eficiente, y las posibilidades serán:
    - INEFFICIENT DRIVING, si ha estado más tiempo entre más de 50% y 75% (incluido).
    - VERY INEFFICIENT DRIVING, si ha estado más tiempo entre más de 75% y 100% (incluido).

Entrada de ejemplo

```
28#10#85#26#50#60##
10#32#100#50#25#150##
STOP TEST
```

Salida de ejemplo

```
EFFICIENT DRIVING
VERY EFFICIENT DRIVING
```

H - Operación “camello”

Nuestra amiga Jenny está preocupada. Le encantan los camellos y siempre ha soñado con viajar al desierto y disfrutar de estos extraordinarios animales. Por esta razón se ha animado a visitar el Medio Oriente en las próximas vacaciones.

El problema se encuentra en que va a hacer mucho calor, y en su destino tendrá que llevar poca ropa, pero se siente un poco acomplejada porque todos sus amigos están en mucha mejor forma, por lo que decide apuntarse al gimnasio para intentar bajar unos kilos antes de sus vacaciones.

Cuando llega al gimnasio quiere saber si su plan de trabajo le permitirá bajar los kilos que necesita antes del día de partida. Para ello, idea una rutina que le permita calcular su avance, y se propone seguir estrictamente dicha rutina. Cada mañana, al levantarse, acudirá al gimnasio donde entrenará muy duro. Al terminar su entrenamiento aprovechará para pesarse y ver lo que ha mejorado desde el día anterior. Como es muy meticulosa sabe que es capaz de mantener la cantidad de peso perdido de cada día, hasta conseguir su objetivo y disfrutar de un viaje en camello en una mejor forma.

Como estamos preocupados por Jenny, vamos a ayudarla con un programa que le permita, a partir de una rutina determinada, y suponiendo que la sigue estrictamente, saber cuántos días tardará en verse en la báscula con el peso deseado. Para ello introducirá en el programa el número de gramos que quiere perder, seguido de los gramos que perderá en cada entrenamiento, y por último el número de gramos que gana el resto del día.

Por ejemplo, imaginamos que los datos de entrada son los siguientes: 2000 600 300

Esto significa que quiere perder 2kg, perdiendo cada entrenamiento 600gr y recuperando el resto del día 300gr. De esta forma, al cabo de 6 días se pesará y habrá perdido al menos los 2000gr que deseaba.

Entrada

Para cada caso de prueba el programa procesará una línea que contendrá 3 números enteros o, p y g. El primer número o representa los gramos que quiere perder, p es lo que pierde tras cada entrenamiento y por último g lo que gana el resto del día.
La entrada finaliza con tres números negativos para los valores de o, p y g.

Salida

Por cada caso de prueba se mostrará el número de días tras los cuales será capaz de verse en la báscula con el peso deseado. Si no es posible alcanzar dicho peso se mostrará el mensaje OLVIDA LOS CAMELLOS.

Entrada de ejemplo

```
2000 600 300
3000 500 300
1000 500 600
-1 -1 -1
```

Salida de ejemplo

```
6
14
OLVIDA LOS CAMELLOS
```