Concurso de Programación
AdaByron 2019

http://www.ada-byron.es

Cuadernillo de problemas

Patrocinado por

accenture

25 AÑOS 15

UAM UNIVERSIDAD AUTONOMA DE MADRID

Realizado en la **Escuela Politécnica Superior (UAM)**
8-9 de marzo de 2019

In almost every computation a great variety of arrangements for the succession of the processes is possible, and various considerations must influence the selections amongst them for the purposes of a calculating engine. One essential object is to choose that arrangement which shall tend to reduce to a minimum the time necessary for completing the calculation.
Ada Byron

8-9 de marzo de 2019

Listado de problemas

Autores de los problemas:

- Carlos Aguirre Maeso (Universidad Autónoma de Madrid)
- José Ramón Dorronsoro Ibero (Universidad Autónoma de Madrid)
- Luis Fernando Lago Fernández (Universidad Autónoma de Madrid)
- Gonzalo Martínez Muñoz (Universidad Autónoma de Madrid)

A
Subiendo escaleras

Los habitantes de Escalera son famosos por su capacidad de subir de golpe varios peldaños de una escalera. Además les gusta especializarse en subir un número determinado de peldaños. Juan, por ejemplo está especializado en subir 1, 3 o 5 peldaños en cada paso (lo que representamos mediante la lista $[1, 3, 5]$). En esta lista de peldaños siempre aparecerá el 1 en primer lugar, pues todos los escaleranos son capaces de subir los peldaños de uno en uno.

Así las cosas nos surge la curiosidad de saber, por ejemplo, de cuántas maneras distintas podría subir una escalera de 10 peldaños un escalerano especializado en subir $[1, 3, 5]$ peldaños. Más generalmente, si un escalerano se especializa en subir $[p_1 = 1, p_2, ..., p_K]$ peldaños, ¿de cuántas maneras diferentes podrá subir una escalera de N peldaños?

Entrada

La entrada está formada por distintos casos de prueba, uno por línea. Cada caso de prueba contiene un primer número entero $1 \leq N \leq 100$ con el número total de peldaños de una cierta escalera. A continuación hay un segundo número entero $1 \leq M \leq 10$ con el número de diferentes peldaños que un escalerano puede subir de golpe. Finalmente hay M números enteros $p_1 = 1, \ldots, p_M$ con los valores de dichos números de peldaños. Se cumple que todos los p_i son distintos entre sí y menores o iguales a N.

El final de las entradas se indica con una línea que empieza con un 0 y que no debe procesarse.

Salida

Para cada caso de prueba se escribirá una línea con el número de diferentes formas en las que el escalerano puede subir la escalera.

Entrada de ejemplo

```
5 2 1 3
5 2 1 4
5 2 1 5
5 3 1 2 3
3 2 1 4
10 3 1 3 5
10 4 1 2 3 4
0
```

Salida de ejemplo

```
4
3
2
13
1
47
401
```

B
Fuga de Alcatraz

Vito y Corleone, dos famosos mafiosos, han decido escaparse de la cárcel de Alcatraz aprovechando que se ha estropeado el sistema de iluminación, que en un descuido la puerta del patio de la prisión se ha quedado abierta y que sobre la isla hay una fuerte niebla. Para lograr escapar deben atravesar un patio rectangular lleno de vigilantes. Al ser la niebla muy cerrada, cada vigilante puede percibir solamente personas que están a una distancia inferior a 10 metros. Las dimensiones del patio son de al menos 100 metros de ancho y 100 metros de largo y como máximo 1000 metros de ancho y 1000 metros de largo. El número de vigilantes en el patio es de al menos 1 y como máximo 1000. Los mafiosos deben decidir si, sabiendo las dimensiones del patio de la prisión, el número de vigilantes que hay y donde están colocados, podrán pasar por el patio sin ser detectados por ningún vigilante.

Entrada

La entrada está formada por distintos casos de prueba. Cada caso de prueba consiste en una única línea. El primer número de cada línea ($A, 100 \leq A \leq 1000$) es un entero que indica la anchura del patio. El segundo número ($L, 100 \leq L \leq 1000$) es un entero que indica la longitud del patio. El tercer número ($N, 1 \leq N \leq 1000$) es un entero que indica la cantidad de vigilantes que hay en el patio. A continuación la línea contendrá N parejas de números enteros. Cada pareja indica la posición (x, y) de un vigilante. Las coordenadas $(0, 0)$ corresponden a la esquina inferior izquierda del patio de la prisión.

El final de la entrada se indica con una línea con un único 0 que no se debe procesar.

Salida

Para cada caso de prueba se escribirá una línea con la palabra `SI` si los dos mafiosos podrían escapar de la prisión sin ser vistos por los vigilantes, o la palabra `NO` si no hay forma de que los mafiosos puedan cruzar el patio sin ser vistos por algún vigilante.

Entrada de ejemplo

```
100 100 1 50 50
100 500 4 80 100 10 10 30 30 60 60
100 100 5 10 10 30 10 50 10 70 10 90 10
0
```

Salida de ejemplo

```
SI
SI
NO
```

C
Combo

En un juego de cartas similar a *Magic* se simula una batalla entre dos poderosos magos. Cada carta representa un ataque o hechizo que se puede lanzar al enemigo, y tiene asociado un coste en *maná*. El *maná* es la energía del mago, que puede usarse para invocar hechizos. El poder de una carta es igual a su coste en *maná*. Así, los ataques más poderosos tienen asociado un mayor coste. Sólo es posible jugar una carta si se dispone de la cantidad suficiente de *maná*.

Además es posible realizar varios ataques a la vez juntando varias cartas en un ataque combinado o *combo*. Para crear un *combo* se aplican las siguientes reglas:

- Una carta se puede considerar un *combo* de tamaño 1, cuyo poder y coste son iguales al *maná* de la misma.
- Se pueden unir dos *combos* gastando una cantidad de *maná* equivalente a la suma de los costes de cada uno de ellos. El poder del *combo* resultante será igual a la suma de los *manás* de todas las cartas incluidas.

Por ejemplo, si un jugador tiene en su mano tres cartas con *maná* 1, 2 y 3 respectivamente, puede hacer las siguientes jugadas:

- Jugar una única carta, con coste/poder 1, 2 o 3.
- Crear un *combo* de dos cartas, con coste/poder 3 (cartas 1 y 2), 4 (cartas 1 y 3), o 5 (cartas 2 y 3).
- Crear un *combo* de tres cartas con un poder de 6. El coste en *maná* de este *combo* dependerá del orden en que se combinan las cartas:
  - Si se añade la carta 3 al *combo* 1-2, el coste total será 9: el coste de crear el *combo* 1-2 (3) más el coste de unir el *combo* 1-2 y la carta 3 (3+3).
  - Si se añade la carta 2 al *combo* 1-3, el coste total será 10: el coste de crear el *combo* 1-3 (4) más el coste de unir el *combo* 1-3 y la carta 2 (4+2).
  - Si se añade la carta 1 al *combo* 2-3, el coste total será 11: el coste de crear el *combo* 2-3 (5) más el coste de unir el *combo* 2-3 y la carta 1 (5+1).

Obviamente, en el caso de querer combinar las 3 cartas, la opción más *barata* es combinar primero la 1 y la 2, y luego añadir la 3. Consideraremos siempre que el coste de crear un *combo* es el menor posible entre todas las opciones.

La pregunta que nos hacemos es, dado un conjunto de cartas y una cantidad de *maná* disponible, ¿cuál es el ataque más potente que se puede realizar? En el ejemplo anterior, con una cantidad de maná de 6, el mejor ataque que se puede hacer es 5. En cambio con una cantidad de maná de 9 o más se podría hacer un ataque de 6.

Entrada

La entrada está formada por distintos casos de prueba. Cada caso de prueba consta de dos líneas. En la primera línea aparecen dos números enteros M y N. El número M ($0 \leq M \leq 500$) es la cantidad de maná disponible. El número N ($1 \leq N \leq 15$) es el número de cartas. En la segunda línea aparecen N números enteros que representan el valor V_i de cada una de las cartas ($1 \leq V_i \leq 20$, $i = 1, 2, ..., N$).

El final de la entrada se indica con una línea con un único cero que no se debe procesar.

Salida

Para cada caso de prueba, se escribirá una línea con el valor del mejor ataque que se puede realizar.

Entrada de ejemplo

```
6 3
1 2 3
10 3
2 1 3
21 5
2 1 3 3 4
0
```

Salida de ejemplo

```
5
6
10
```

D
Haciendo largos

El médico me ha recomendado nadar para cuidar mi espalda, así que desde esta semana estoy yendo a la piscina. Al ser mis primeros días estoy haciendo series cortas, de sólo dos largos, y descanso un rato entre series. Pero como estoy en muy baja forma, necesito hacer los descansos cada vez más largos. Así, en mi primer día de natación mi primer descanso fue de un minuto, el segundo de un minuto y medio, el tercero de dos minutos y así sucesivamente. Cada descanso duró 30 segundos más que el anterior y, como todos los largos los hice en el mismo tiempo (30 segundos), tardé 37 minutos en nadar 20 largos.

Así las cosas, me pregunto de manera general cuánto tardaría en hacer un determinado número de largos en función del tiempo por largo, el número de largos por serie y los tiempos de descanso entre series.

Entrada

La entrada está formada por distintos casos de prueba, y cada caso de prueba consiste en una línea con 5 números enteros: N, T, M, D e I. El primer entero, N, es el número de largos que quiero nadar. El segundo número, T es el tiempo en segundos que tardo en hacer un largo. El tercer entero, M, es el número de largos que hago en cada serie. Finalmente D indica el tiempo en segundos del primer descanso entre series, e I es la cantidad extra de descanso, en segundos, que necesito con cada serie adicional. Nótese que la última serie no tiene por qué estar completa.

Se satisfacen los siguientes límites: $1 \leq N \leq 1000000$, $20 \leq T \leq 40$, $1 \leq M \leq N$, $0 \leq D \leq 60$, $0 \leq I \leq 60$.

El final de la entrada se indica con una línea con un único cero que no se debe procesar.

Salida

Para cada caso de prueba, se escribirá una línea con el tiempo en segundos necesario para hacer los largos.

Entrada de ejemplo

```
20 30 2 60 30
20 30 2 60 60
0
```

Salida de ejemplo

```
2220
3300
```

E
CamelCasi

En programación es habitual utilizar el estilo *camel case* para escribir los nombres de variables que concatenan varias palabras. La norma de estilo indica que la primera letra de cada una de las palabras concatenadas debe ser mayúscula, mientras que el resto deben escribirse en minúscula. Algunos ejemplos de nombres de variables utilizando esta notación son `CamelCase`, `AdaByron` o `ConcursoDeProgramacion`.

En sus prácticas de programación, Jaime ha decidido utilizar el estilo *camel case* para nombrar todas sus variables. Sin embargo ha cometido algunos errores al teclear, y algunas de las letras que deberían ser mayúsculas aparecen en minúscula en su programa. Como consecuencia el compilador ha mostrado un montón de errores, y Jaime ha decidido llamar a su estilo *camel casi*.

¿Puedes ayudar a Jaime a arreglar su programa?

Entrada

La entrada consiste en una lista de variables en formato *camel casi*. Cada variable aparece en una línea distinta, y puede haber repeticiones. Se garantiza que ninguna variable tiene una longitud mayor de 50 caracteres y que todas las variables aparecen al menos una vez bien escritas (en formato *camel case* correcto) en la lista. Además no existen dos variables que se diferencien sólo en el caso. Por ejemplo las cadenas `AdaByron` y `adaByron` se refieren a la misma variable. El máximo número de variables en la lista es 10000.

Salida

Para cada variable de entrada en formato *camel casi* se imprimirá una línea con el nombre de la variable en formato *camel case*.

Entrada de ejemplo

```
camelcase
AdaByron
ConcursodeProgramacion
CamelCase
concursoDeprogramacion
camelCase
ConcursoDeProgramacion
adabyron
Adabyron
```

Salida de ejemplo

```
CamelCase
AdaByron
ConcursoDeProgramacion
CamelCase
ConcursoDeProgramacion
CamelCase
ConcursoDeProgramacion
AdaByron
AdaByron
```

F
Pintando la pared

Para pintar una pared de su habitación, Jorge ha decidido seguir el siguiente procedimiento: Moja el rodillo en pintura, lo coloca horizontalmente en un punto arbitrario de la pared, y lo hace rodar hasta llegar al suelo. Después de un rato pintando de este modo, la zona pintada de la pared es un conjunto de rectángulos solapados que llegan hasta el suelo de la habitación, y Jorge se pregunta cuál será su área total.

Entrada

La entrada está formada por distintos casos de prueba, y cada caso de prueba consiste en una única línea con varios números. El primer número de cada linea es un entero A, $(1 \leq A \leq 5000)$ que indica la anchura del rodillo. El siguiente número es un entero N $(1 \leq N \leq 1000)$ que indica el número de veces que se pasa el rodillo por la pared. A continuación aparecen N parejas de números enteros (x_i, y_i), $i = 1, ..., N$. Cada pareja indica las coordenadas en la pared en las que se sitúa el lado izquierdo del rodillo en cada una de las pasadas. Las coordenadas $x = 0$, $y = 0$ corresponden a la esquina inferior izquierda de la pared. Se cumplen las condiciones $0 \leq x_i \leq 20000$ y $1 \leq y_i \leq 1000$.

El final de la entrada se indica con una línea con un único 0 que no se debe procesar.

Salida

Para cada caso de prueba, se escribirá una línea con un número entero igual al área de la región pintada de la pared.

Entrada de ejemplo

```
10 2 10 20 15 10
10 3 10 5 12 20 13 10
0
```

Salida de ejemplo

```
250
220
```

G
La madriguera del señor Conejo

El señor Conejo y su familia tienen una madriguera muy particular. Para poder refugiarse rápidamente cuando aparece el señor Zorro, cada conejo entra en la madriguera por un agujero distinto. Estos agujeros están situados a lo largo de una línea recta y, por supuesto, están hechos a la medida exacta de cada conejo. Tanto es así que cuando celebran una fiesta y comen más de la cuenta, los conejos no son capaces de salir por el agujero que les corresponde. En estas situaciones cada conejo tiene que buscar un agujero más grande que el suyo para poder salir, y normalmente se arma un follón considerable.

Así que han decidido organizarse de la siguiente manera: Para entrar a la madriguera, cada uno lo hace por su agujero. Para salir, todos se mueven hacia la derecha, y cuando encuentran el primer agujero más grande que el suyo salen. Al final de la madriguera han construido un agujero lo suficientemente grande como para que todos puedan salir por él si es necesario. El último conejo siempre tiene que salir por este agujero.

La pregunta que nos hacemos es: Dada una serie de agujeros, ¿por cuál de ellos sale cada conejo?

Entrada

La entrada está formada por distintos casos de prueba, cada uno en una línea distinta. En cada línea aparece en primer lugar un número entero N ($1 \leq N \leq 10000$), el número de conejos y agujeros. A continuación aparece una lista con N números enteros d_1, d_2,..., d_N, que representan los diámetros de cada uno de los N agujeros por los que entran los conejos. Los diámetros de todos los agujeros satisfacen $1 \leq d_i \leq 20$.

El final de la entrada se indica con una línea con un único 0 que no se debe procesar.

Salida

Para cada caso de prueba, se escribirán en una línea los índices de los agujeros por los que sale cada conejo. El primer agujero de la madriguera tiene el índice 0, y el último, de tamaño mayor que todos los demás, el índice N.

Entrada de ejemplo

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

Salida de ejemplo

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

H
La serie de Leibnitz

La serie de Leibnitz es una serie infinita cuyo valor converge a $\pi/4$:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots = \frac{\pi}{4}$$

Aunque puede utilizarse para calcular el valor de π, en la práctica la convergencia es muy lenta y es necesario sumar muchos términos para lograr una precisión de unos pocos decimales. Por ejemplo, sumando el primer millón de términos de la serie se obtiene un valor de π igual a $3.1415916535\ldots$, que sólo es exacto hasta la quinta cifra decimal.

El objetivo de este ejercicio es calcular la suma de los primeros N términos de la serie de Leibnitz con un determinado número M de posiciones decimales. Es importante calcular cada uno de los términos de la suma con el número de decimales requerido antes de sumarlo. Nótese que el número obtenido de este modo puede ser distinto del resultado de truncar a M decimales la suma de los mismos N términos calculados con un número de cifras decimales mayor.

Por ejemplo, la suma de los 100 primeros términos de la serie, calculados con 10 decimales, da como resultado:

```
0.7828982264
```

En cambio sumando los mismos 100 términos, calculados con 50 decimales, y truncando el resultado final a 10 decimales, se obtiene:

```
0.7828982259
```

Para $N = 100$ y $M = 10$ el número que se desea obtener es el primero.

Entrada

La entrada está formada por distintos casos de prueba, cada uno descrito por dos números enteros. El primer entero, N, es el número de términos de la serie que se quiere sumar. El segundo entero, M, es el número de cifras decimales que se desea tener. Se garantiza que $1 \leq N \leq 20000$ y $0 \leq M \leq 1000$.

El final de la entrada se indica con una línea con un único 0 que no se debe procesar.

Salida

Para cada caso de prueba se escribirá en una línea el valor de la suma de los N primeros términos de la serie de Leibnitz con M cifras decimales.

Entrada de ejemplo

```
100 10
100 50
100 0
1 12
2 7
0
```

Salida de ejemplo

```
0.7828982264
0.782898225889638191076853559569230129100763596101 64
1.
1.000000000000
0.6666667
```

I
Repartidor de frigoríficos

Un repartidor de frigoríficos debe decidir en qué cruce de calles situar su local, de modo que la distancia total de todos los trasportes sea mínima. El repartidor trabaja en la ciudad de Novapolis donde todas las calles son en cuadrícula. Las distancias más cortas entre dos puntos se deben por tanto calcular siguiendo las calles y no como la distancia euclídea. La dificultad añadida es que en la furgoneta del repartidor solo cabe un frigorífico por lo que después de cada reparto debe volver al local a cargar el siguiente frigorífico.

A continuación se muestra un ejemplo con una cuadrícula 3×3 en el que los asteriscos indican los lugares de reparto (posiciones $(1,1)$, $(1,3)$ y $(3,1)$). En este caso la mejor localización para el local del repartidor será el punto $(1,1)$.

```
  1  2  3
1 *--|--*
2 |--|--|
3 *--|--|
```

Entrada

La entrada está formada por distintos casos de prueba. Cada caso de prueba consiste en varias líneas donde la primera línea son tres números enteros: C, el número de calles en dirección vertical, F, el número de calles en dirección horizontal y N, el número de frigoríficos a repartir. A continuación aparecen N líneas con dos números enteros cada una que indican la posición (x, y) de cada sitio de reparto. Se satisface que $1 \leq x \leq C$, $1 \leq y \leq F$, $1 \leq F, C \leq 1000$ y $1 \leq N \leq 10000$.

El final de la entrada se indica con una línea con tres ceros que no se debe procesar.

Salida

Para cada caso de prueba, se escribirán en una línea las coordenadas (i, j) de la posición óptima del local del repartidor. En caso de existir varias posiciones óptimas, se escribirán las coordenadas de aquella tal que la suma $i + j$ sea mínima. Si aún así hubiera varias opciones, se debe considerar la posición con menor i.

Entrada de ejemplo

```
10 10 2
1 1
10 10
15 15 7
1 12
2 11
3 10
4 4
10 3
11 2
12 1
0 0 0
```

Salida de ejemplo

```
1 1
4 4
```

J
Teclado de televisor

Queremos buscar series y películas en la televisión de la forma más eficiente posible. Cuando se busca un texto en la televisión aparece un teclado virtual por el que nos desplazamos con las flechas del mando a distancia: arriba, derecha, abajo e izquierda. El objetivo es calcular el número mínimo de desplazamientos que hay que realizar dado un teclado y un texto a introducir determinados. Debe tenerse en cuenta que hay teclas de distinto tamaño y que se trata de un teclado circular: cuando el cursor se sale por un extremo del teclado, aparece por el extremo opuesto. El cursor se situa inicialmente sobre la tecla que ocupa la esquina superior izquierda.

A continuación se muestra un ejemplo de teclado:

```
AAABCD
AAAGHI
LEEEEK
```

Un teclado tiene forma rectangular, y todas sus teclas son también rectángulos, aunque puede haber teclas de diferentes tamaños. Así, en el teclado del ejemplo la tecla A tiene dimensiones 2×3, la tecla B tiene dimensiones 1×1, la E 1×4, etc. El efecto de pulsar cada una de las flechas del mando a distancia se describe a continuación:

- Derecha: el cursor se desplaza a la tecla de la derecha, y si hay varias a la de más arriba. En el ejemplo, si se pulsa *derecha* cuando el cursor está sobre la tecla A, el cursor se situará sobre la tecla B.
- Izquierda: el cursor se desplaza a la tecla de la izquierda, y si hay varias a la de más arriba. En el ejemplo, si se pulsa *izquierda* cuando el cursor está sobre la tecla A, el cursor se situará sobre la tecla D.
- Arriba: el cursor se desplaza a la tecla de arriba, y si hay varias a la de más a la izquierda. En el ejemplo, si se pulsa *arriba* cuando el cursor está sobre la tecla E, el cursor se situará sobre la tecla A.
- Abajo: el cursor se desplaza a la tecla de debajo, y si hay varias a la de más a la izquierda. En el ejemplo, si se pulsa *abajo* cuando el cursor está sobre la tecla A, el cursor se situará sobre la tecla L.

Para el teclado del ejemplo, una posible forma de teclar el texto ADKEAB con el mínimo número de desplazamientos podría ser: izquierda, arriba, izquierda, arriba y derecha.

Entrada

La entrada está formada por la descripción de una serie de teclados, y hay varios textos de prueba para cada teclado. La descripción de un teclado empieza con una línea con tres números enteros: F, el número de filas del teclado; C, el número de columnas; y N, el número de textos a introducir usando el teclado. Se garantiza que $1 \leq F \leq 8$, $1 \leq C \leq 20$ y $1 \leq N \leq 100$. A continuación aparecen F líneas, con C caracteres cada una, describiendo el teclado según el formato del ejemplo anterior. Los caracteres válidos van desde el *espacio* (ASCII= 32) hasta la Z (ASCII=90). Las teclas son siempre rectangulares y nunca hay dos teclas distintas asociadas al mismo carácter.

Tras la descripción del teclado aparecen N líneas, cada una con una cadena de texto a introducir usando el teclado. Todas las cadenas están formadas con los caracteres asociados a las teclas del teclado, y nunca

tienen una longitud mayor que 100.

El final de la entrada se indica con una línea con tres ceros que no se debe procesar.

Salida

Para cada texto de prueba se debe imprimir, en una línea distinta, el número mínimo de desplazamientos necesarios para teclearlo.

Entrada de ejemplo

```
3 6 4
AAABCD
AAAGHI
LEEEEK
ADKEA
BCD
ELE
A
1 4 2
ABBC
ABC
CBA
0 0 0
```

Salida de ejemplo

```
4
3
4
0
2
3
```