

Eres un experto en ciberseguridad y llevas años trabajando en el sector. Sin embargo, la vida no se ha portado bien contigo y las deudas y los problemas son cada vez mayores. Todo va mal y la desesperación crece dentro de ti. Ya llevas un tiempo pensando en usar tus habilidades en algo, se podría decir que en algo poco ético, atracar un banco. Robar todo el dinero posible y huir a un país extranjero, alejado de preocupaciones, sonaba perfecto. Tu sueño es atracar un banco al puro estilo de La Casa de Papel.

Por si te has olvidado aquí te dejamos un resumen de cómo ejecutar y compilar tu programa en Linux:

| Lenguaje de programación | Compilar programa | Ejecutar programa |
| --- | --- | --- |
| C / C++ | g++ -o program program.cpp | ./program <1.in >1.out |
| Java | javac program.java | java program <1.in >1.out |
| Python | - | python program.py <1.in >1.out |

Hackear cuentas-A

Acabas de lograr infiltrarte en el sistema de seguridad del banco central. Para acceder al sistema principal, necesitas conocer la suma total de las claves de seguridad de los empleados que tienen acceso a los sistemas del banco y cuyas claves superan un valor X. Por lo tanto, deberás sumar todas las claves de seguridad de cada empleado y, en caso de que la suma total sea mayor que X, añadirla a la cuenta.

Como eres un experto has conseguido obtener permisos de root en el sistema y recuperar una lista con las claves de todos los empleados.

¡Buena suerte!

ENTRADA

La entrada representará la lista de claves de seguridad y consistirá en un número indefinido de líneas organizadas de la siguiente manera: la primera línea contiene el valor X que deberán superar las claves, el resto de las líneas contiene cada una todas las claves que tiene un empleado.

Ten en cuenta que la lista de claves de seguridad para cada empleado terminará con una línea en blanco, y que la lista completa también terminará con una línea en blanco.

SALIDA

La salida consistirá en la suma total de las claves de seguridad de los empleados, cuyas claves superen el valor X.

*EJEMPLO ENTRADA*

2000

20

30

100

90

200

1000

20

30

900

10

2000

10

9000

10

1

1

1

1

1000

1000

*EJEMPLO SALIDA*

13024

*EXPLICACIÓN*

La suma de las claves del primer empleado es igual a 1440, que al no ser mayor que 2000, no se añade a la cuenta.

Las claves del segundo y del tercer empleado tampoco suman esa cantidad, por lo que no se añaden.

Las claves del cuarto empleado suman 2010, así que se añaden al total

Las del quinto suman 9000, que también es añadido al total.

Las del último suman 2014, por lo que se añade al total.

Finalmente, si sumamos 2010 + 9000 + 2014, obtenemos el resultado: 13024

Contando bancos-B

Tus cálculos dicen que el dinero ideal para vivir el resto de tu vida sin preocupaciones es una cifra D. Para ello, has decidido atracar una serie de bancos hasta reunir ese dinero.

Como no quieres terminar en la cárcel, has investigado todos los bancos de la ciudad para hacerte una idea de cuánto dinero puedes conseguir con el atraco, A, y cuál es la posibilidad de que te pille la policía, P.

Además, como necesitas el dinero rápido y preparar cada golpe requiere mucho tiempo, también quieres saber el número mínimo de bancos que necesitas atracar de cada nivel de peligro.

Tu objetivo es desarrollar un programa que, dados un número, N, de conjuntos (A, P), devuelva el máximo valor de peligro que tienes que asumir para robar, al menos, el dinero que necesitas, D, y cuantos bancos de cada nivel debes atracar.

ENTRADA

La entrada consiste en un número indeterminado de casos.

Cada caso empieza con una línea con dos números enteros positivos, que representan el número de bancos (N <= 10000) y el dinero que necesitas respectivamente (D <= 100000).

A continuación, habrá N líneas que contendrán dos números enteros positivos, que representarán los valores A (100 <= A <= 10000) y P (1 <= P <= 100), respectivamente.

Un caso con N = 0 y D = 0 indica el final de los datos.

SALIDA

La salida consiste en imprimir para cada caso 'Caso i:', donde i representa el número de caso empezando desde 1, seguido con una línea por cada nivel de peligro, ordenados de menor a mayor, y los bancos que tienes que atracar en el formato:

'P -> Nº bancos'. En el caso en que no se pueda llegar al mínimo de dinero se imprimirá 'imposible'.

Debes dejar una línea en blanco entre cada caso.

*EJEMPLO ENTRADA*

5 1000

50 1

150 1

400 2

320 3

1500 5

7 500

200 2

300 1

50 1

10 1

1000 3

500 8

140 1

1 100

50 4

0 0

*EJEMPLO SALIDA*

* Caso 1:

5 -> 1

* Caso 2:

1 -> 4

* Caso 3:

Imposible

*EXPLICACIÓN*

Primer caso:

La suma de los 4 primeros bancos es 5 + 150 + 400 + 320 = 920, por lo que necesitamos atracar el último banco. Como el dinero que obtenemos del último banco es mayor que necesitamos, solo atracando ese sería suficiente.

Segundo caso:

Puedes conseguir el dinero atracando solo bancos de peligro 1.

Tercer caso:

No puedes conseguir suficiente dinero.

El banco perfecto-C

Quieres atracar uno de los muchos bancos de la ciudad. Casualmente todos ellos se encuentran en una única avenida en el centro. Eres muy previsor, y sabes que en cuanto entres sonará la alarma y todos los policías de la calle serán notificados de inmediato, entonces quieres elegir un banco tal que se maximice el tiempo que tarda en venir la policía.

Esta calle al ser tan céntrica solo tiene bancos ('B'), comisarías de policía ('P'), y tiendas ('T'); la policía tarda siempre 1 minuto en recorrer una manzana.

A pesar de ser notificados de inmediato, la policía tardará 10 minutos adicionales en prepararse y planear una estrategia, lo que te da un poco de margen.

ENTRADA

La primera línea contiene un entero T (1 <= T <= 1000), el número de casos de prueba. Siguen T casos de prueba, cada uno compuesto de dos líneas.

Para cada caso, la primera línea contiene un entero N (2 <= N <= 100000), el número de manzanas en la calle. La siguiente línea contiene N caracteres sin espacios entre ellos, 'B' representa un banco, 'P' representa una comisaría de policía y 'T' representa una tienda.

Para cada caso, siempre existe al menos un banco y una comisaría de policía en la calle, por lo tanto, siempre hay solución.

Se tiene que (2 <= T\*N <= 100000000).

SALIDA

Para cada caso un único entero por línea, el tiempo en minutos que tarda en venir la policía si eliges el banco óptimo.

*EJEMPLO ENTRADA*

4

3

PTB

5

PBTTP

4

PBBP

6

PBBBPP

*EJEMPLO SALIDA*

12

11

11

12

*EXPLICACIÓN*

En el primer caso: La policía se prepara 10 minutos antes de salir, y después tiene que ir de la posición 0 a la 2 (la primera comisaría de policía es la posición 0), que son 2 minutos. 12 minutos en total.

En el segundo caso: La comisaría en la posición 0 se prepara y responde en 1 minuto.

En el tercer caso: Si atracas el banco en la posición 1, llegará la policía de la posición 0; si atracas el banco en la posición 2, llegará la policía de la posición 4.

En el cuarto caso: Si atracas cualquiera de los bancos en las posiciones 1 o 4, la policía se prepara y llegará en 1 minuto, si atracas el banco de la posición 2, ambos policías tardarán 2 minutos en llegar.

La caja fuerte-D

Tras varios meses analizando la caja de seguridad del banco, por fin, has descubierto cómo obtener el código correcto:

En la puerta de la caja aparecerán N números enteros positivos, que pueden variar su valor en cualquier momento. Dados estos números, el sistema te pedirá calcular el valor mínimo entre

dos posiciones un número indeterminado de veces y, si respondes correctamente a todas antes de que se acabe el tiempo, se abrirá la caja fuerte.

A primera vista parece sencillo, pero tras muchos intentos te has dado cuenta de que es imposible responder dentro del tiempo. Por eso, te has propuesto crear un programa que te permita obtener ese valor mínimo en el menor tiempo posible.

ENTRADA

La entrada consiste en un número indefinido de casos. Cada caso empieza con una línea indicando el número de elementos en el array, N (N <= 200000). Las siguientes N líneas contendrán un entero, comprendido entre 0 y 1000, que indicará el valor inicial del elemento.

Luego se presentarán hasta 200000 líneas, donde cada una representa una acción, hay 3 tipos:

- "S x r" cambia el valor del elemento en la posición x a r.

- "M x y" calcula el mínimo entre x e y.

- "END" Fin del caso.

Un caso con N = 0 indica el fin de la entrada.

SALIDA

Para cada caso se produce una línea 'Caso n:', donde n es el número del caso empezando por 1.

Para cada “M x y” se mostrará una línea conteniendo el mínimo.

Deja una línea en blanco entre casos.

*EJEMPLO DE ENTRADA*

3

100

100

100

M 1 1

M 1 3

S 2 200

M 1 2

S 3 0

M 2 3

END

10

1

2

3

4

5

6

7

8

9

10

M 1 10

END

0

*EJEMPLO SALIDA*

Caso 1:

100

100

100

0

Caso 2:

1

La banda-E

Ya has decidido cuál será tu objetivo, así que es hora de empezar a organizar el gran golpe. La primera tarea es buscar compañeros.

Como no estás muy metido en este mundillo y no sabes con quienes puedes contar, has decidido llamar a un conocido que tiene contactos y te ha pasado un listado de profesionales, de los que algunos ya han trabajado juntos.

Para facilitar el golpe estás buscando tener un grupo lo más grande posible, pero tampoco quieres arriesgarte a que varios integrantes se lleven mal y eso pueda perjudicar el atraco. Por ello has decidido contratar a aquellos que ya han trabajado juntos y has supuesto que, si A y B, y B y C trabajan bien juntos, A y C también trabajarán bien juntos.

El objetivo es buscar el grupo con el mayor número de personas que trabajan bien juntas.

ENTRADA

La entrada consiste en varios datasets. La primera línea contiene un número indicando los casos que hay.

La primera línea de cada caso contiene dos números N y M, donde N es el número de atracadores y M las parejas que han trabajado juntos. Las siguientes M líneas contendrán dos enteros A y B que significa que A y B han trabajado juntos. Puede haber líneas repetidas.

SALIDA

La salida consta de un número indicando cuánta gente hay en el grupo más grande de personas para cada caso.

*EJEMPLO ENTRADA*

2

3 2

1 2

2 3

10 12

1 2

3 1

3 4

5 4

3 5

4 6

5 2

2 1

7 1

1 2

9 10

8 9

*EJEMPLO SALIDA*

3

7

Los guardias de seguridad-F

Si no queréis acabar en la cárcel no se os puede escapar ningún detalle. Habéis estudiado a los guardias de seguridad que van a trabajar en el banco ese día y les habéis asignado un valor de peligro. Algunos de los guardias tienen un valor negativo, ya que han sido sobornados previamente para obtener información, y cooperarán con vosotros. Los guardias se organizan por turnos, y habéis podido haceros con los turnos de los guardias de seguridad el día del atraco. Vuestro objetivo es encontrar para cada turno el valor máximo de peligro.

ENTRADA

Habrá T casos (1 <=T <= 1000) compuestos por dos líneas cada uno. La primera línea de cada caso contiene un entero N (1 <= N <= 1000). La segunda línea contien N enteros, A (-10⁹ <= A <= 10⁹)

SALIDA

Se mostrará el resultado de cada turno en una línea con el formato “Caso X: P”, donde X representa el número de turno y P el nivel máximo de peligro.

*EJEMPLO ENTRADA*

2

8

1 -1 10 1 2 0 2 -5

4

1 1 2 -3

*EJEMPLO SALIDA*

Caso 1: 16

Caso 2: 4

Cajeros-G

Para maximizar las ganancias en el atraco también vais a robar el dinero de todos los cajeros automáticos que tengáis tiempo a abrir. En el banco hay distintos tipos de cajeros, todos ellos tienen distintas cantidades de dinero y tiempo en el que se tarda en abrirlos.

Dado un tiempo límite y dos arrays en los que uno de ellos indica la cantidad de dinero (expresado en miles de euros) en el cajero y el otro el tiempo el que se tarda en abrirlo (la posición en el array es el número del cajero). Debéis calcular la máxima cantidad de dinero que puedes conseguir en el tiempo dado.

ENTRADA

La entrada consiste en tres líneas. La primera indica el tiempo límite L (1 <= L <= 30) y el numero de cajeros N (1 <= N <= 100) respectivamente. La segunda contiene un array de N elementos expresando el dinero D (0 <= D <= 1000) que contiene cada cajero. La tercera línea contiene un array de N elementos indicando el tiempo T (0 <= T <= 50) necesario para abrir cada cajero.

SALIDA

La salida consistirá en tres líneas donde se indicarán el tiempo que se tarda en abrir los cajeros, un array con los cajeros que se han de abrir para conseguir la cantidad máxima de dinero posible y el dinero conseguido, respectivamente.

*EJEMPLO ENTRADA*

20 7

10 30 25 32 68 54 23

02 05 10 10 09 10 08

*EJEMPLO SALIDA*

19

5 6

122

Cámaras-H

Para evitar que la policía os identifique durante el atraco habéis decidido hackear el sistema de seguridad de las cámaras del banco. Para ello, habéis desarrollado un método basándoos en la suposición de que todas las contraseñas son capicúa y generadas a partir de una semilla, N.

En uno de tus análisis has conseguido robar las semillas de varios empleados del banco para poder vulnerar sus contraseñas. Para hacerlo más cómodo decidís desarrollar un pequeño programa que lo haga por vosotros y así poder dedicar vuestro tiempo a otras cosas.

Las reglas que debe seguir el programa para poder generar las contraseñas a partir de una semilla son las siguientes:

* Todas las contraseñas tienen la misma cantidad de dígitos que N (la semilla), son positivas y no pueden ser iguales a N.
* Si la contraseña inmediatamente posterior a N tiene una cantidad diferente de dígitos, el proceso debe reanudarse desde la semilla, pero siguiendo en la dirección opuesta.

ENTRADA

La entrada consistirá en un número indefinido de líneas, cada una de las cuales contiene lo siguiente: primero una contraseña, N, la semilla a partir de la cual se generan las otras contraseñas, luego un número M que representa la cantidad de contraseñas a generar, y finalmente, un número X que especifica la dirección en la que debes calcular las contraseñas, siendo 1 para las contraseñas mayores que N y -1 para las inferiores a N.

SALIDA

El resultado mostrará, para cada línea, una línea con las contraseñas generadas separadas por un espacio en blanco.

*EJEMPLO ENTRADA*

343 2 -1

565 1 1

111 3 -1

*EJEMPLO SALIDA*

333 323

575

101 121 131

A por el botín-I

¡Tras mucho esfuerzo, hemos conseguido acceder a la cámara acorazada! Ahora viene la parte más importante del golpe, robar el botín.

El botín está organizado en forma de matriz de N x N joyas, y, tras mucha investigación, hemos descubierto que para evitar las alarmas debemos robar las joyas de forma que sus posiciones formen un sub-rectángulo dentro de la matriz.

Cada joya tiene asociado un valor, que puede ser negativo si la joya es falsa o muy difícil de vender.

Tu tarea consiste en calcular el sub-rectángulo en el que la suma de todos sus valores sea la máxima posible.

ENTRADA

La entrada consistirá en una primera fila con un valor entero positivo N (N <= 100), seguida de N filas de N enteros separados por un espacio en blanco.

SALIDA

La salida será la suma del máximo sub-rectángulo

*EJEMPLO ENTRADA*

4

 0 -2 -7 0

 9 2 -6 2

-4 1 -4 1

-1 8 0 2

EJEMPLO SALIDA

15

EXPLICACIÓN:

0  -2  -7  0

9   2  -6  2

     -4   1  -4  1

     -1   8   0  2

El máximo sub-rectángulo es:

9 2

       −4 1

       −1 8

Cuya suma es 15