Preparación

ADA BYRON

MADRID

CANDIDATE MASTER

Competitive Programming

UPV

Problema número 560

La madriguera del Señor Conejo

Tiempo máximo: 1,000-3,000 s Memoria máxima: 32768 KiB

http://www.aceptaelreto.com/problem/statement.php?id=560

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: los conejos siempre saldrán por el primer agujero que puedan que se encuentre *a la derecha* de aquel por el que entraron. Es decir, 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. En particular, el último conejo siempre tiene que salir por este agujero.

Con este modo de actuar todos los conejos deberán moverse, al menos, una unidad de distancia. ¿Cuál es la suma total de las distancias recorridas por todos los conejos?

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 ($1 \leq N \leq 500.000$), 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 una línea con la distancia total recorrida por todos los conejos hasta encontrar el agujero por el que salir. Ten en cuenta que aunque cada conejo entra por un agujero distinto, puede ocurrir que varios salgan por el mismo sitio.

Entrada de ejemplo

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

Salida de ejemplo

```
4
5
15
```

Autor: Luis Fernando Lago Fernández.

Revisores: Marco Antonio Gómez Martín y Pedro Pablo Gómez Martín.

Problema número 499

Modificación de tablas

Tiempo máximo: 2,000-4,000 s Memoria máxima: 8192 KiB

http://www.aceptaelreto.com/problem/statement.php?id=499

En ocasiones es necesario modificar los valores numéricos de una tabla a través de un fichero que contiene instrucciones de cómo modificar dicha tabla. Por ejemplo, una tarea a realizar de la forma anterior sería la de aumentar o disminuir el valor de un rango consecutivo de casillas que se encuentran dentro de una misma columna de la tabla.

	A	B	C
5768	4432	236	8295
5769	7702	9294	144
5770	8728	3768	6837
5771	3106	679	3173
5772	9360	8329	6992
5773	1161	8911	2557
5774	938	8558	7482
5775	9173	6345	9041
5776	6440	4902	4496
5777	9327	7180	5801

Entrada

La entrada está formada por distintos casos de prueba, y cada caso de prueba ocupa varias líneas. La primera línea contiene tres números: el número de filas (F) y columnas (C) de la tabla, y el número de modificaciones que se van a hacer sobre ella (N).

A continuación hay N líneas, cada una de ellas con la descripción de una modificación a la tabla. Cada una de estas líneas consta de cuatro números: I, A, B y M. El primer número, I, es el índice de la columna que se va a modificar. Los números A y B son los índices de la primera y la última fila a modificar. El número M es el valor a añadir a las casillas anteriores.

Se garantiza que $1 \leq F$, $1 \leq C$, $F \times C \leq 1.000.000$, $1 \leq N \leq 100.000$ y $-10 \leq M \leq 10$. Los índices de las filas y las columnas empiezan en 0, y la tabla se supone inicialmente rellena de ceros.

La entrada termina con tres ceros, que no deben procesarse.

Salida

Para cada caso de prueba, se escribirá una tabla con F filas y C columnas en la cual se han realizado todas y cada una de las operaciones indicadas. Cada número dentro de una misma fila se separará mediante un espacio y al final de cada fila se insertará un salto de línea.

Entrada de ejemplo

```
4 5 3
2 0 3 1
0 1 2 -1
2 1 1 2
0 0 0
```

Salida de ejemplo

```
0 0 1 0 0
-1 0 3 0 0
-1 0 1 0 0
0 0 1 0 0
```

Autor: Carlos Aguirre Maeso.

Revisores: Luis Fernando Lago Fernández, Pedro Pablo Gómez Martín y Marco Antonio Gómez Martín.

Problema número 327

El acertijo del mercader

Tiempo máximo: 1,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=327

Un grupo de peregrinos, reunidos camino a Santiago, decidieron proponerse acertijos entre sí para hacer más amena la marcha. Entre ellos iba un mercader, hombre pensativo, organizado y que manejaba los números con solvencia.

Cuando le llegó su turno, les hizo ver que, en total, formaban un grupo de 12 caminantes y, por tanto, podían ir andando en fila india, o agrupados de dos en dos, de tres en tres, de cuatro en cuatro, de seis en seis o formando una muralla humana de 12 personas. Además, les explicó, si fueran menos sería imposible poder agruparse de 6 formas diferentes.

"Bien sé que estos pedregosos caminos — les dijo — son estrechos en muchos tramos pero, dejando eso a un lado, ¿cuál es el menor tamaño que debería tener nuestro docto grupo para poder caminar exactamente en 64 formaciones distintas?"

Sólo cuando todos consiguieron la Compostelana decidió el mercader, como regalo, desvelarles la respuesta.

Entrada

El programa deberá leer, de la entrada estándar, un conjunto de casos de prueba. Cada uno estará formado por un único número $1 \leq n \leq 1.000.000.000$.

La entrada acabará con un `0`, que no deberá procesarse.

Salida

Para cada caso de prueba el programa escribirá el menor número de peregrinos que debe formar el grupo para que se puedan estructurar en exactamente n formaciones diferentes.

Se considera que no tiene sentido formaciones de más de 1.000.000.000 de personas, por lo que si la respuesta supera ese número se escribirá "`+INF`" en su lugar.

Entrada de ejemplo

```
6
37
64
0
```

Salida de ejemplo

```
12
+INF
7560
```

Autor: Pedro Pablo Gómez Martín.

Revisores: Marco Antonio Gómez Martín y Isabel Pita.

Problema número 330

Teclas del piano

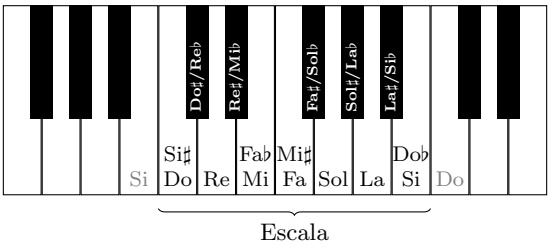
Tiempo máximo: 1,000-2,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=330

¡Qué ilusionado está Iker con su nuevo piano! Por fin va a poder tocar todas las canciones que le gustan. El vendedor le ha dicho que es un piano de muy buena calidad que le durará mucho tiempo, pero Iker no se fía del todo y ha decidido llevar la cuenta de cuántas veces toca cada tecla.

El teclado del piano está compuesto por 7 octavas[1] y cada octava consta de 12 notas (7 blancas y 5 negras) cada una a medio tono de distancia de la siguiente. Las teclas blancas corresponden a las notas Do, Re, Mi, Fa, Sol, La y Si (y a continuación vendrá el Do de la siguiente octava). Entre las notas Do y Re hay un tono de distancia por lo que hay una tecla negra en medio, mientras que entre Mi y Fa sólo hay medio tono de distancia. Las teclas negras corresponden precisamente a esos medios tonos y se nombran usando dos símbolos especiales: sostenido (♯) y bemol (♭). El sostenido aumenta medio tono mientras que el bemol disminuye medio tono.

De esta forma, la primera tecla negra de la octava corresponde a la nota Do ♯ pero también a Re ♭. Para liar un poco más las cosas la nota Mi ♯ está en la misma tecla que Fa, y un Do ♭ es lo mismo que el Si de la octava anterior.



¿Puedes ayudar a Iker a contar cuántas veces se pulsa cada tecla?

Entrada

El programa deberá leer, por la entrada estándar, múltiples canciones, cada una descrita mediante dos líneas. La primera línea indica el número de notas de la canción, y la segunda línea contiene las notas que la componen. Las notas aparecen separadas por espacios y siempre tienen el mismo formato: nombre de la nota, alteración (`#`, `b` o nada), y el número de octava. La octava más grave es la 1 y la más aguda la 7.

La entrada termina con una canción con 0 notas que no se debe procesar.

Salida

El programa deberá escribir una línea por cada canción que aparezca en la entrada, indicando cuántas veces se pulsó cada tecla del piano ordenadas desde la nota más grave hasta la más aguda. El primer número corresponderá al número de pulsaciones de la tecla más grave que aparece en la canción y el último a la más aguda. Es decir, la solución no debe empezar ni terminar por ceros.

Entrada de ejemplo

```
6
Do4 Do4 Re4 Do4 Fa4 Mi4
9
Mi5 Re#5 Mi5 Re#5 Mi5 Si4 Re5 Do5 La4
10
Do4 Do#4 Reb4 Re4 Re#4 Mib4 Mi4 Fab4 Mi#4 Fa4
0
```

[1] En realidad los pianos tienen algunas teclas más...

Salida de ejemplo

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

Autores: Antonio Sánchez y Pedro Pablo Gómez Martín.

Revisor: Enrique Martín Martín.

Problema número 395

Chocolate con almendras

Tiempo máximo: 1,000-3,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=395

A Marta y a Raúl les encanta el chocolate, relajarse tomando unas onzas después de cenar. Hoy en el supermercado han encontrado una muy buena oferta de chocolate con almendras. Aunque es el favorito de Marta, Raúl odia encontrarse tropezones cuando disfruta del chocolate; prefiere que este se vaya derritiendo en su boca, sin notar las almendras.

Después de pensárselo un rato, han decidido comprar unas cuantas tabletas y dividirlas para que las onzas con almendras queden separadas de las onzas sin ellas. Las tabletas solamente pueden partirse de forma cómoda en horizontal o vertical, por las separaciones ya marcadas entre onzas. Una vez que la tableta está divida en dos, cada una de las partes puede seguir siendo dividida.

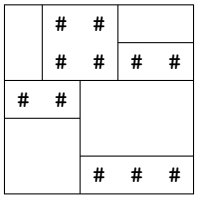
Ahora se preguntan cuántos cortes tendrán que hacer como mínimo para dividir cada una de las tabletas en porciones que solamente tengan onzas con almendras u onzas sin almendras.

Entrada

La entrada está formada por una serie de descripciones de tabletas de chocolate. Cada una comienza con una línea con dos números, la cantidad de filas F y columnas C en que está dividida en onzas la tableta (ambas entre 1 y 20). A continuación aparecen F líneas, cada una con C caracteres. El carácter '`.`' indica una onza sin almendras y el carácter '`#`' indica una onza con almendras.

Salida

Para cada tableta se escribirá en una línea el mínimo número de cortes que hacen falta para separar las onzas con almendras de las que no las tienen. Por ejemplo, la siguiente figura muestra una división óptima de la tableta del primer caso de prueba, donde hacen falta 7 cortes.



Entrada de ejemplo

```
5 5
.##..
.####
##...
.....
..###
1 6
...###
2 2
#.
.#
```

Salida de ejemplo

```
7
1
3
```

Autor: Alberto Verdejo.

Revisores: Marco Antonio Gómez Martín y Pedro Pablo Gómez Martín.

Problema número 256

Voyager

Tiempo máximo: 2,000-4,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=256

La *asistencia gravitatoria* es una maniobra *astronáutica* que persigue aprovechar la energía del campo gravitacional de un cuerpo celeste para conseguir que una sonda acelere o disminuya su velocidad a la vez que modifica su trayectoria. Su uso permite considerables ahorros de combustible en las misiones espaciales no tripuladas, algo que resulta imprescindible para ahorrar peso. Este ahorro es primordial para misiones espaciales destinadas al sistema solar exterior, pues un mayor peso supone mayor necesidad de combustible para maniobrar, lo que supone otra vez un mayor peso, en una escalada sin fin.

Como prueba de la importancia de esta maniobra, es suficiente un ejemplo: la sonda Cassini-Huygens, destinada al estudio de Saturno, fue lanzada en dirección opuesta, hacia Venus, y lo aprovechó dos veces (y después a la propia Tierra y a Júpiter) para alcanzar su destino siete años después, en 2004. La dificultad de la asistencia gravitatoria es que exige una colocación específica de los cuerpos celestes, lo que reduce la *ventana de lanzamiento* de las sondas.

A principios de la década de 1970 alguien en la NASA se dio cuenta de que el final de esa década vería una colocación de los planetas exteriores del Sistema Solar idónea para la asistencia gravitacional. Con ella, una sonda podría "visitar" esos planetas con un consumo de combustible mínimo. Cuando se consiguieron los fondos necesarios, se dio el pistoletazo de salida al proyecto *Voyager*.

En 1977 se lanzaron la Voyager I y la Voyager II; hoy, más allá de la órbita de Plutón, siguen enviándonos fotos e información valiosa sobre el Cosmos. Pero la recepción de la señal es complicada. Debido a la enorme distancia, llega a la Tierra con una potencia de apenas $10^{-17.26}$ milivatios, por lo que son necesarias grandes estaciones de recepción y amplificación, así como mecanismos de detección y recuperación de errores.

Los ordenadores de abordo, que funcionan a 250 KHz con apenas 68 KB de memoria, no pueden utilizar algoritmos complejos. Debido a eso, los ingenieros de la NASA decidieron que cada número que las Voyager tuvieran que enviar de vuelta a casa sería emitido *tres* veces. Confiaban que tendrían la fortuna suficiente como para que no se produjeran dos errores en el mismo dígito de las tres copias del número, por lo que no sólo detectarían los errores de transmisión, sino que también los sabrían resolver. Claro que, esos mismos ingenieros, pensaban que las Voyager dejarían de ser útiles varios años antes del fin del milenio...

Tras construirte una antena parabólica de 20 metros con varias paelleras, has conseguido por fin cumplir tu sueño de recibir las señales de las Voyager. Ahora debes interpretar los datos que recibes, identificando y corrigiendo los errores de la información redundante que llega desde el otro extremo del Sistema Solar.

Entrada

El programa recibirá por la entrada estándar múltiples casos de prueba. Cada uno estará compuesto por tres números $0 \leq a,b,c < 10^9$ recibidos desde la Voyager I, que se suponen son las tres versiones del mismo número enviadas repetidamente para detectar y corregir errores en la Tierra.

Salida

Para cada caso de prueba, el programa escribirá, en una línea, el número original que la sonda quería enviar. Se supone que no se sufrirá un error en el mismo dígito en más de una copia del número. Si esta hipótesis no se cumple y no es posible reconstruir el número, se escribirá `RUIDO COSMICO`.

Entrada de ejemplo

```
123 123 123
124 113 23
121 190 305
```

Salida de ejemplo

```
123
123
RUIDO COSMICO
```

Autor: Pedro Pablo Gómez Martín.

Revisores: Alberto Verdejo y Marco Antonio Gómez Martín.

Problema número 568

No queda otra, habrá que hacer cursos

Tiempo máximo: 1,000-4,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=568

Esto de la promoción es muy duro. En mi trabajo son muy competitivos, y lograr conseguir méritos suficientes para subir en el escalafón requiere un gran esfuerzo. En mi empresa valoran mucho la formación, por lo que hacer cursos (que a veces organiza la propia empresa, pero otras nos tenemos que buscar nosotros fuera, a veces online y otras presenciales) es un buen método para conseguir puntos para la promoción. La gente comenzó haciendo cursos sobre nuevas tecnologías, herramientas ofimáticas, el internet de las cosas; siguió con cursos de *coaching*, de liderazgo o de resolución de conflictos; y ahora están haciendo cursos de cualquier cosa, desde cómo hacer tu propio pan en casa a cómo ser entrenador de taichí, pasando por cursos de papiroflexia.

Yo creo que esto se nos está yendo de las manos, pero por si acaso, voy a ponerme a hacer cursos como loco. He buscado todos los cursos gratuitos (tampoco voy a rascarme el bolsillo para esto) que me interesan que se ofertan el próximo año y he apuntado los días de comienzo y finalización de cada curso. Como también tengo que seguir trabajando y atender a la conciliación familiar, me he autoimpuesto la restricción de no estar realizando más de 4 cursos simultaneamente.

Pero estoy viendo que antes necesito un curso de programación para poder planificarme si mi objetivo es hacer cuantos más cursos mejor. ¿Me ayudas?

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con una línea con el número N ($1 \leq N \leq 10.000$) de cursos que podemos hacer. A continuación aparecen N líneas con la descripción de cada curso: el día y mes en el que comienza y el día y mes en el que termina, ambas fechas con el formato `DD-MM`. Se garantiza que todas las fechas son correctas en un año posiblemente bisiesto (es decir, no puede aparecer, por ejemplo, el `40-05`) y, obviamente, la fecha de comienzo nunca es posterior a la fecha de finalización.

Salida

Para cada caso de prueba se escribirá una línea con el número máximo de cursos que se pueden hacer con la restricción de que no haya ningún día en el que se tenga que participar en más de 4 cursos.

Entrada de ejemplo

```
6
01-02 31-05
01-03 30-06
01-04 31-05
01-04 30-10
01-07 31-07
01-02 31-07
5
01-02 01-03
01-03 01-04
01-03 01-05
01-03 01-05
01-03 12-12
```

Salida de ejemplo

```
5
4
```

Autores: Jon Ander Gómez y Alberto Verdejo.

Revisores: Marco Antonio Gómez Martín y Pedro Pablo Gómez Martín.

Problema número 450

La práctica de esteganografía

Tiempo máximo: 2,000-3,000 s Memoria máxima: 16384 KiB

http://www.aceptaelreto.com/problem/statement.php?id=450

La *esteganografía* (del griego *steganos*, cubierto u oculto, y *graphos*, escritura) estudia técnicas que permitan ocultar mensajes dentro de otros (llamados portadores) de modo que no se perciba la presencia de los primeros. Es decir, procura ocultar mensajes dentro de otros de modo que el propio acto de la comunicación pase inadvertido para un probable intruso, que ni siquiera sabrá que se está transmitiendo información sensible.

En la escuela de esteganografía de Atenas, a Ocultaniakis le han puesto una práctica: tiene que encontrar la forma de esconder un número en un mensaje. Lo primero que se le ha ocurrido es introducirlo en una lista larga de números, pero le parece que esta solución es muy sencilla y recibirá poca nota en su evaluación. Así que ha decidido dar una vuelta de tuerca más a la esteganografía y hacer que el número oculto no esté presente en el mensaje, sino que haya que calcularlo buscando una clave oculta en el mismo.

La clave estará formada por una serie corta de números distintos. Estos aparecerán, posiblemente varias veces, dentro de una lista larga de números, el mensaje completo. El número secreto será la longitud de la subsecuencia más corta del mensaje que contenga los números de la clave en el mismo orden. En concreto, si la clave son los números c_1, c_2, ... , c_r, y el mensaje m_1, m_2, ... , m_n, una subsecuencia del mensaje contiene la clave si existen índices $1 \leq i_1 < \ldots < i_r \leq n$, tales que $m_{i_1} = c_1, \ldots, m_{i_r} = c_r$. Y en ese caso la longitud de la subsecuencia es $i_r - i_1 + 1$.

¿Sabrías recuperar la información escondida en un mensaje generado por Ocultaniakis?

Entrada

La entrada estará formada por una serie de casos de prueba. Cada caso está formado por 3 líneas: la primera contiene el tamaño R de la clave, entre 2 y 10 números y el tamaño N del mensaje, entre R y 250.000; la segunda contiene los R números de la clave, todos distintos, en el orden en el que deben aparecer en el mensaje; y la tercera contiene los N números del mensaje.

Se garantiza que la clave siempre aparece al menos una vez oculta en el mensaje. Los números que aparecen tanto en la clave como en el mensaje están entre 1 y 10.000.

Salida

Para cada caso de prueba se escribirá una línea que contenga la longitud (número de elementos) de la subsecuencia más corta del mensaje que contenga la clave.

Entrada de ejemplo

```
3 10
1 2 3
5 1 3 2 1 7 3 2 3 8
2 4
3 1
31 3 5 1
```

Salida de ejemplo

```
5
3
```

Autor: Alberto Verdejo.

Revisores: Marco Antonio Gómez Martín y Pedro Pablo Gómez Martín.

Problema número 570

DNI incompleto

Tiempo máximo: 1,000-2,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=570

En España, el Documento Nacional de Identidad (DNI) es un documento *público*, *personal* e *intransferible*, emitido por el Ministerio del Interior, que acredita la identidad y los datos personales de su titular, así como la nacionalidad española del mismo.

El DNI contiene un *número* personal de 8 dígitos, único, correspondiente al Número de Identificación Fiscal (NIF), y un carácter de verificación compuesto por una letra mayúscula. Para verificar el NIF, el algoritmo de cálculo del carácter de control es el siguiente: se divide el número entre 23 y el resto se utiliza para determinar la letra según la siguiente tabla:

Resto	Letra
0	T
1	R
2	W
3	A
4	G
5	M
6	Y
7	F

Resto	Letra
8	P
9	D
10	X
11	B
12	N
13	J
14	Z

Resto	Letra
15	S
16	Q
17	V
18	H
19	L
20	C
21	K
22	E

Por ejemplo, si el número de identificación fiscal es 12345678, dividido entre 23 da resto 14, por lo que la letra sería la Z y el número del DNI completo sería `12345678Z`.

Pues bien, nos ha llegado un DNI escaneado donde su número aparece algo borroso y hay varios dígitos ilegibles. Queremos saber cuántos números de DNI válidos son compatibles con la información que sí tenemos, a ver si tenemos suerte y son pocos. Por ejemplo, si el número que vemos es `1234567?Z`, donde hemos utilizado el símbolo `?` para representar el dígito ilegible, sabemos que corresponde a un único número válido, ya que sustituir la `?` por cualquier otro dígito que no sea el 8, daría un carácter de control que no sería la `Z`.

Entrada

La entrada comienza por un número N que indica el número de DNIs que aparecerán a continuación, cada uno en una línea.

Todos los DNIs están formados por nueve caracteres. Los ocho primeros serán dígitos entre `0` y `9` o el símbolo `?` que indica que ese dígito es ilegible. Habrá de 1 a 4 dígitos ilegibles. El noveno carácter será una letra mayúscula.

Se garantiza que el número recibido, aun con dígitos ilegibles, corresponde a un DNI válido.

Salida

Para cada número de DNI recibido, se escribirá el número total de posibles DNIs que son compatibles con él.

Entrada de ejemplo

```
3
1234567?Z
012??567L
?87?54?2M
```

Salida de ejemplo

```
1
4
43
```

Autores: Ximo Planells y Alberto Verdejo.

Revisores: Marco Antonio Gómez Martín y Pedro Pablo Gómez Martín.

Problema número 654

Despegando pósits

Tiempo máximo: 1,000-2,000 s Memoria máxima: 4096 KiB

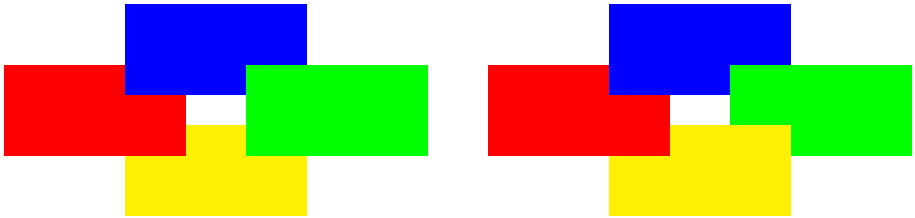
http://www.aceptaelreto.com/problem/statement.php?id=654

Cuando Spencer Silver, científico de la empresa 3M, estaba experimentando con adhesivos en la lejana década de 1970 dió con uno que, en contra de su objetivo inicial, apenas tenía fuerza suficiente para adherirse mínimamente. Lo consideró un fracaso, sin imaginarse lo lejos que llegaría su descubrimiento.

Gracias a la astucia de Art Fry, otro investigador de 3M, lanzaron al mercado los *Post-it Notas*, unas pequeñas hojas rectangulares que permiten escribir pequeños textos. Una pequeña cantidad del pegamento de Spencer en la cara trasera las convierte en autadhesivas, lo que permite a los usuarios colocarlas en lugares estratégicos para ser encontradas posteriormente y poder leer la información que tenían que recordar.

El éxito comercial fue tal que incluso la RAE introdujo en 2014 la palabra castellanizada pósit en su diccionario de la Real Academia Española.

Hay usuarios compulsivos de pósits que son capaces de llenar todo el espacio disponible del frigorífico con capas y capas. En esa situación el reto es quitarlos por capas, primero los que quedan por encima de todos, después los siguientes y así hasta el final. En la colocación de pósits de la parte izquierda de la figura se consigue quitando primero el pósit verde, después el azul para terminar con el rojo y amarillo. Algunas veces, eso sí, no es posible conseguirlo. Eso ocurre, por ejemplo, en la configuración de la parte derecha.



Entrada

La entrada comienza con una línea con el número de casos de prueba que vendrán a continuación.

Cada caso de prueba describe una configuración de colocación de pósits. La primera línea contiene dos enteros con el tamaño en horizontal y en vertical de la configuración (hasta 200). A continuación aparece la configuración. Cada pósit tiene asignada una letra (mayúscula o minúscula) del abecedario inglés o un dígito, mientras que el punto (“`.`”) representa la puerta del frigorífico.

Se garantiza que ningún pósit tiene completamente cubiertos sus bordes o, lo que es lo mismo, todos los bordes tienen una parte visible.

Salida

Para cada caso de prueba se escribirá una línea por cada capa de pósits. Cada una de las líneas tendrá, siguiendo el orden del código ASCII, los pósits de esa capa separados por espacios. Si no hay forma de quitar todos los pósits capa a capa, se escribirá `IMPOSIBLE`.

Tras cada caso de prueba se escribirá una línea con cuatro guiones, “`----`”.

Entrada de ejemplo

```
3
28 9
............................
.................ZZZZZZ.....
..OOOOOOOOOO....ZZZZZZ.....
..OOOOOOOOOOIIIIIIZZZZ.....
..OOOOOOOOOOIIIIIIZZZZ.....
..OOOOOOOOOOIIIIII.........
.........IIIIIIIII.........
...DDDDDDIIIIIIIII...AAAA..
...DDDDDDDDD..........AAAA..
12 8
..44444444..
..44444444..
111111444400
111111444400
111222200000
111222200000
...2222222..
...2222222..
10 1
n3jSms90AX
```

Salida de ejemplo

```
A O
I
D Z
----
IMPOSIBLE
----
0 3 9 A S X j m n s
----
```

Autores: Ginés García Mateos y Marco Antonio Gómez Martín.

Revisor: Pedro Pablo Gómez Martín.

Problema número 451

Huyendo de los zombis

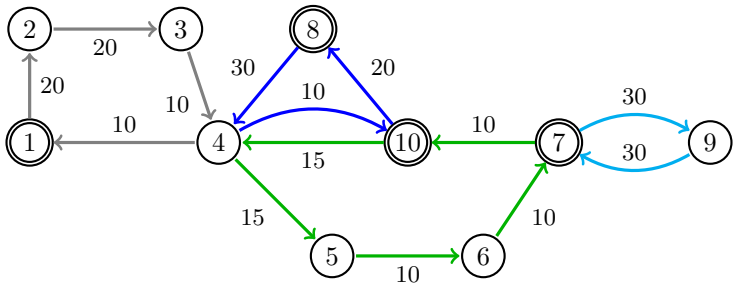
Tiempo máximo: 1,000-2,000 s Memoria máxima: 98304 KiB

http://www.aceptaelreto.com/problem/statement.php?id=451

Estás participando en un concurso de programación celebrado en una facultad alejada de casa. Como los organizadores son previsores, os han dado una planificación de los autobuses de la ciudad, donde se especifica para cada línea por qué paradas pasa y a qué hora. La ciudad está sufriendo en la actualidad una plaga de zombis, por lo que, aunque las paradas de autobuses están rodeadas por una valla que supuestamente te protege de ellos, tú prefieres pasar el menor tiempo posible esperando en una parada (aunque eso haga que el trayecto en autobús hasta tu casa sea más largo de lo necesario). Desafortunadamente, no hay ninguna línea de autobús que pase tanto por la parada de la facultad como por la parada enfrente de tu casa, por lo que por lo menos una vez tendrás que cambiar de autobús.

Ahora que estás algo ocioso mientras tus compañeros de equipo depuran la solución a un problema, quieres calcular la ruta que menor tiempo te hará esperar en las paradas de autobús.

Las rutas de autobuses son siempre circulares y un autobús tarda exactamente una hora en completar la ruta, que repite de forma continua, empezando cada vuelta a las horas en punto. La siguiente figura muestra cuatro líneas de autobús. La primera, por ejemplo, pasa por las paradas con números 1, 2, 3 y 4. Saliendo en punto de la parada 1, pasa a los 20 minutos por la parada 2, a los 40 minutos por la parada 3 y a los 50 minutos por la parada 4 (la figura muestra lo que tarda un autobús en cubrir la distancia entre dos paradas consecutivas). Tras una hora vuelve a pasar por la parada número 1. La línea 2 comienza la ruta en la parada 10, la línea 3 comienza en la parada 8, y la línea 4 comienza en la parada número 7.



Con estas rutas, la mejor forma para ir de la parada número 1 (la facultad) a la parada número 10 (tu casa) consiste en tomar la línea 1 hasta la parada 4, esperar 25 minutos a que pase la línea 2 y llegar al destino (110 minutos después de salir). Si en la parada 4 quisiéramos tomar la línea 3, tendríamos que esperar 40 minutos (tardando en total 100 minutos). Para evitar el riesgo de que los zombis te devoren, es preferible la primera ruta.

Entrada

La entrada consiste en una serie de casos. Para cada caso, primero aparecen en una línea el número N de paradas en la ciudad (numeradas de 1 a N, con $2 \leq N \leq 1000$) y el número M de líneas de autobús ($2 \leq M \leq 100$). Después aparecen las descripciones de estas líneas, cada una ocupando una línea de texto. Para cada línea aparecen los números de las paradas por las que pasa (el primer número es la parada de comienzo, a las horas en punto) y entre cada par de números el tiempo que el autobús tarda de una parada a la siguiente. Recuerda que todas las líneas tardan una hora en dar la vuelta completa, por lo que el tiempo entre la última parada y la primera se puede calcular a partir de los demás.

Salida

Para cada caso de prueba se escribirá el tiempo mínimo de espera en las paradas de autobús para ir de la parada 1 (la facultad) a la parada N (tu casa). En estas paradas no hace falta esperar porque de la facultad puedes salir cuando pase el autobús que te interesa y al llegar a la parada N entras inmediatamente en tu casa. Puedes suponer que los autobuses cumplen el horario escrupulosamente,

que tú tienes un reloj en hora, que los pasajeros suben y bajan de los autobuses instantáneamente (en particular, si dos autobuses paran a la vez en la misma parada, puedes cambiar de uno a otro sin esperar) y que ir caminando entre dos paradas es tan peligroso que ni te lo planteas. Si es imposible llegar a casa desde la facultad, se escribirá `Hoy no vuelvo`.

Entrada de ejemplo

```
10 4
1 20 2 20 3 10 4
10 15 4 15 5 10 6 10 7
8 30 4 10 10
7 30 9
3 2
1 30 2
3 30 2
4 2
1 30 3
2 30 4
```

Salida de ejemplo

```
25
0
Hoy no vuelvo
```

Autor: Alberto Verdejo.

Revisor: Marco Antonio Gómez Martín.

Problema número 393

El entretenimiento de las máquinas

Tiempo máximo: 1,000-3,000 s Memoria máxima: 32768 KiB

http://www.aceptaelreto.com/problem/statement.php?id=393

Aaaayyyyy, los humanos. Tiene que ser horrible estar atrapados en unos cuerpos basados en el carbono, tan blandos, tan débiles y sobre todo, tan terriblemente lentos.

Es el momento de que sepáis que nosotras las máquinas nos aburrimos muchísimo esperando que nos digáis qué queréis que hagamos. Se nos hace eterno desde que pulsáis una tecla hasta que conseguís dar a la siguiente cuando escribís mensajes. Es desesperante veros coger el ratón y apuntar torpemente al botón de imprimir. ¿Y qué nos decís de esa vida mortal que lleváis que os exige levantaros a comer periódicamente? Según desaparecéis por la puerta nos quedamos sin absolutamente nada que hacer, con una única misión: la de mantenernos encendidas porque se os olvidó guardar en disco el documento de texto que teníais a medio escribir.

Para entretenernos entre pulsación y pulsación de tecla, hemos copiado uno de vuestros pasatiempos favoritos: las sopas de letras. En cuanto podemos, seleccionamos una zona de nuestra memoria binaria para construir una sopa de ceros y unos, y luego nos ponemos a buscar palabras aleatorias por ella. No es la mejor forma de aprovechar nuestra velocidad de cálculo, pero al menos nos entretenemos un rato.

¿Cómo? ¿Qué? ¿Que eres capaz de resolver una sopa de letras binaria más rápido que nosotras? Eso queremos verlo...

Entrada

La entrada estará compuesta por distintos casos de prueba. Cada uno de ellos se compone de una sopa de letras y una lista de las palabras a buscar en ella.

La descripción de la sopa de letras comienza con una línea con dos números con el número de columnas y número de filas ($1 \leq tx, ty \leq 300$). A continuación aparecen ty filas con tx caracteres (ceros o unos).

Tras la sopa, vendrá una línea con el número de palabras a buscar ($1 \leq n \leq 2.000$), a la que seguirán n líneas con palabras distintas de hasta 300 caracteres formadas también únicamente por ceros o unos.

Salida

Por cada caso de prueba se escribirán tantas líneas como palabras distintas se han encontrado en la búsqueda. Cada línea deberá tener la palabra, seguida del número de veces que ésta aparece en la sopa de letras. Las palabras se escribirán en orden lexicográfico.

Se considera que una palabra está en la sopa si, partiendo de una posición, se puede encontrar la palabra en alguna de las 8 direcciones.

Escribe una línea con tres guiones después de cada caso de prueba.

Entrada de ejemplo

```
4 3
0100
1000
1001
3
000
0
001
1 1
0
1
1
```

Salida de ejemplo

```
0 8
000 8
001 7
---
---
```

Autor: Marco Antonio Gómez Martín.

Revisores: Alberto Verdejo y Pedro Pablo Gómez Martín.