Olimpiada interuniversitaria
Área de Tecnología
Octubre de 2017

Instrucciones: A continuación se le presenta una serie de problemas, cada problema tiene una valoración en puntos, debe tratar de realizar programas de computadora que resuelvan cada problema, puede resolver los problemas en cualquier orden deseado, al finalizar de resolver cada problema debe solicitar que sea validado con el archivo de prueba que le será entregado por los jueces, deberá generar su salida y entregarla para su verificación, si la verificación es correcta habrá obtenido los puntos en que se ha valorado el problema. Recuerde que el tiempo utilizado para resolver los problemas también es parte de la competencia. A menos que se indique otro método, los problemas deberán solicitar el nombre del archivo de entrada y generar la salida a un archivo nombrado salidaN.txt, donde N corresponde al número de problema.

Problema No. 1 (X pts.)

El informático holandés Edsger Dijkstra hizo muchas contribuciones importantes al campo, incluyendo el algoritmo de búsqueda de caminos más corto que lleva su nombre. Este problema no se trata de ese algoritmo.

En un examen de algoritmos se le quitó un punto por escribir de forma incorrecta "Dijkstra" - entre D y stra, escribió un número de caracteres, cada uno de los cuales era i, j o k. Usted está dispuesto a recuperar el punto, para esto usará quaternions, un sistema numérico real (extendido desde números complejos) con la siguiente estructura multiplicativa:

	1	***i***	***j***	***k***
1	1	*i*	*j*	*k*
i	*i*	-1	*k*	*-j*
j	*j*	*-k*	-1	*i*
k	*k*	*j*	*-i*	-1

Para multiplicar un quaternion por otro, mire la fila para el primer quaternions y la columna para el segundo quaternion. Por ejemplo, para multiplicar i por j, busque en la fila para i y la columna para j para encontrar que la respuesta es k. Para multiplicar j por i, mire la fila para j y la columna para i para encontrar que la respuesta es -k.

Como se puede ver en los ejemplos anteriores, los quaternions no son conmutativos, es decir, hay algunos **a** y **b** para los cuales **a * b! = B * a**. Sin embargo, son asociativos - para cualquier **a**, **b**, **c**, es cierto que **a * (b * c) = (a * b) * c**.

Los signos negativos antes de los quaternions funcionan como lo hacen normalmente - para cualquier quaternion **a** y **b**, es cierto que **-a * -b = a * b**, y **-a * b = a * -b = - (a * b)**.

Usted quiere argumentar que su error ortográfico era equivalente a la ortografía correcta *ijk* mostrando que puede dividir su cadena de **i**s, **j**s y **k**s en dos lugares, formando tres subcadenas, de modo que la subcadena más a la izquierda reduce (en multiplicación de quaternion) a ***i*** , la subcadena media se reduce a ***j***, y la subcadena derecha se reduce a ***k***. (Por ejemplo, *jij* sería

interpretado como *j * i * j; j * i* es *-k*, y *-k * j* es *i*, por lo que *jij* se reduce a *i*.) Si esto es posible, obtendrá su punto. ¿Puede encontrar una manera de hacerlo?

Entrada:
La primera línea del archivo de entrada contiene el número de casos de prueba, **T**. **T** casos de prueba siguen uno por línea. Cada uno consta de una línea con dos enteros separados por espacio **L** y **X**, seguido por otra línea con **L** caracteres, todos los cuales son **i, j** o **k**. Tenga en cuenta que la cadena nunca contiene signos negativos, 1s o cualquier otro caracter. La cadena que se va a evaluar es la cadena dada de L caracteres repetidos X veces. Por ejemplo, para **L = 4, X = 3**, y la cadena dada **kiij**, su cadena de entrada sería **kiijkiijkiij**.

Salida:
Para cada caso de prueba, la salida de una línea que contiene "Caso # x: y", donde **x** es el número de caso de prueba (a partir de 1) y **y** es **SI** o **NO**, dependiendo de si la cadena se puede dividir en tres partes que reducen a i, j y k, en ese orden, como se ha descrito anteriormente.
Ejemplo:

Entrada	Salida
5 2 1 ik 3 1 ijk 3 1 kji 2 6 ji 1 10000 i	Caso #1: NO Caso #2: SI Caso #3: NO Caso #4: SI Caso #5: NO

Problema No. 2 (X pts.)

Tic-Tac-Toe-Tomek es un juego jugado en un tablero cuadrado de 4 x 4. El tablero comienza vacío, excepto que un solo símbolo de "T" puede aparecer en uno de los 16 cuadrados. Hay dos jugadores: X y O. Se turnan para hacer movimientos, con X comenzando. En cada movimiento un jugador pone su símbolo en uno de los cuadrados vacíos. El símbolo del jugador X es 'X', y el símbolo del jugador O es 'O'.
Después del movimiento de un jugador, si hay una fila, columna o una diagonal que contenga 4 de los símbolos de ese jugador, o que contenga 3 de sus símbolos y el símbolo 'T', gana y el juego termina. De lo contrario, el juego continúa con el movimiento del otro jugador. Si todos los campos están llenos de símbolos y nadie ganó, el juego termina en empate. Vea la entrada de muestra para ejemplos de varias posiciones ganadoras.
Dada una descripción de 4 x 4 tablas que contiene 'X', 'O', 'T' y '.' (donde '.' representa un cuadrado vacío), que describe el estado actual de un juego, determina el estado del juego Tic-Tac-Toe-Tomek. Los estados a elegir son:

1. "X ganó" (el juego terminó y X ganó)
2. "O ganó" (el juego ha terminado y O ha ganado)
3. "Draw" (el juego ha terminado, y terminó en un empate)
4. "El juego no ha terminado" (el juego aún no ha terminado)

Si hay celdas vacías, y el juego no ha terminado, debe salir "Juego no ha terminado", incluso si el resultado del juego es inevitable.

Entrada
La primera línea de la entrada da el número de casos de prueba, **T**. **T** casos de prueba siguen. Cada caso de prueba consta de 4 líneas con 4 caracteres cada una, con cada carácter siendo **'X', 'O', '.'** o **'T'** (comillas para mayor claridad solamente). Cada caso de prueba es seguido por una línea vacía.

Salida
Para cada caso de prueba, la salida de una línea que contiene "Caso # x: y", donde **x** es el número de caso (a partir de 1) e **y** es uno de los estados dados anteriormente. Asegúrese de obtener los estados exactamente correctos. Cuando ejecuta el código en la entrada de muestra, debe crear la salida de muestra exactamente, incluyendo el "Caso # 1:", la letra mayúscula "O" en lugar del número "0", y así sucesivamente.

Ejemplo

Entrada	Salida
6 XXXT .... OO.. .... XOXT XXOO OXOX XXOO XOX. OX.. .... OOXX OXXX OX.T O..O XXXO ..O. .O.. T...	Caso #1: X gano Caso #2: Draw Caso #3: El juego no ha terminado Caso #4: O gano Caso #5: O gano Caso #6: O gano

```
OXXX
XO..
..O.
...O
```

Problema No. 3 (X pts.)

A John le gustan los palíndromos, y piensa que son justos (lo cual es una palabra elegante para agradar). Un palíndromo es sólo un número entero que lee lo mismo hacia atrás y hacia delante - así que 6, 11 y 121 son todos palíndromos, mientras que 10, 12, 223 y 2244 no son (aunque 010 = 10, no consideramos ceros a la hora de determinar si un número es un palíndromo).
Recientemente se interesó también por los cuadrados, y formó la definición de un número justo y cuadrado - es un número que es un palíndromo y el cuadrado de un palíndromo al mismo tiempo. Por ejemplo, 1, 9 y 121 son justos y cuadrados (siendo palíndromos y cuadrados, respectivamente, de 1, 3 y 11), mientras que 16, 22 y 676 no son justos y cuadrados: 16 no es un palíndromo, 22 no es un cuadrado, y mientras 676 es un palíndromo y un número cuadrado, es el cuadrado de 26, que no es un palíndromo.
Ahora quiere buscar números justos y cuadrados más grandes. Su tarea es, dado el intervalo que John está buscando, decirle cuántos números justos y cuadrados hay en el intervalo, así él sabrá cuándo los ha encontrado todos.

Entrada
La primera línea de la entrada da el número de casos de prueba, las líneas **T**. **T** líneas siguen. Cada línea contiene dos enteros, **A y B** - los extremos del intervalo que John está buscando.

Salida
Para cada caso de prueba, la salida de una línea que contiene "Caso # x: y", donde x es el número de caso (a partir de 1) e y es el número de números justos y cuadrados mayor o igual a **A** y menor o igual a **B**.

Ejemplo:

Entrada	Salida
3 1 4 10 120 100 1000	Caso #1: 2 Caso #2: 0 Caso #3: 2

Problema No. 4 (X pts.)

Siguiendo un viejo mapa, usted ha tropezado con el tesoro secreto de Larry, el Pirata del Pavor.
El tesoro se compone de N armarios cerrados, cada uno de los cuales sólo se puede abrir con una llave de un tipo específico. Además, una vez que se utiliza una llave para abrir un cofre, nunca se puede usar de nuevo. Dentro de cada cofre, por supuesto, encontrará un montón de tesoros, y también puede encontrar una o más llaves que puede utilizar para abrir otros cofres. Un cofre puede contener varias llaves del mismo tipo, y puede contener cualquier número de

llaves.

Usted ya tiene al menos una llave y su mapa dice qué otras llaves se pueden encontrar dentro de los diversos cofres. Con toda esta información, ¿puede averiguar cómo desbloquear todos los cofres?

Por ejemplo, supongamos que el tesoro se compone de cuatro cofres como se describe a continuación, y usted comenzó con exactamente una llave de tipo 1:

Numero de Cofre	Tipo de llave que abre el cofre	Tipo de llave dentro del cofre
1	1	Ninguna
2	1	1,3
3	2	Ninguna
4	3	2

Puede abrir todos los cofres en este ejemplo si los hace en el orden 2, 1, 4, 3. Si comienza abriendo primero el cofre # 1, habrá usado su única llave y quedará atrapado.

Entrada

La primera línea de la entrada da el número de casos de prueba, T. T casos de prueba siguen. Cada caso de prueba comienza con una sola línea que contiene dos enteros positivos K y N, que representan el número de llaves con las que empieza y el número de cofres que necesita para abrir.

Esto es seguido por una línea que contiene K enteros, que representa los tipos de las claves con las que se inicia.

Después de eso, habrá N líneas, cada una representando un solo cofre. Cada línea comenzará con los números enteros Ti y Ki, indicando el tipo de llave necesaria para abrir el cofre y el número de llaves dentro del cofre. Estos dos enteros serán seguidos por Ki más enteros, indicando los tipos de las llaves contenidas dentro del cofre.

Salida

Para cada caso de prueba, la salida de una línea que contiene "Caso #x: C1 C2 ... CN", donde x es el número de casos (a partir de 1), y donde Ci representa el índice (a partir de 1) deberías abrir

Si hay múltiples formas de abrir todos los cofres, elige la forma "lexicográficamente más pequeña". En otras palabras, debe elegir hacer C1 tan pequeño como sea posible, y si hay varias formas de hacer C1 lo más pequeño posible, elija el que haga que C2 sea lo más pequeño posible, y así sucesivamente.

Si no hay manera de abrir todos los cofres, en su lugar debería dar salida a una línea que contenga "Caso #x: IMPOSIBLE".

Ejemplo

Input	Output
3 1 4 1 1 0 1 2 1 3 2 0 3 1 2 3 3	Caso #1: 2 1 4 3 Caso #2: 1 2 3 Caso #3: IMPOSIBLE

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

Problema No. 5 (27 pts.)

Roller coasters are so much fun! It seems like everybody who visits the theme park wants to ride the roller coaster. Some people go alone; other people go in groups, and don't want to board the roller coaster unless they can all go together. And *everyone* who rides the roller coaster wants to ride again. A ride costs 1 Euro per person; your job is to figure out how much money the roller coaster will make today.

The roller coaster can hold **k** people at once. People queue for it in groups. Groups board the roller coaster, one at a time, until there are no more groups left or there is no room for the next group; then the roller coaster goes, whether it's full or not. Once the ride is over, all of its passengers re-queue in the same order. The roller coaster will run **R** times in a day.

For example, suppose **R**=4, **k**=6, and there are four groups of people with sizes: 1, 4, 2, 1. The first time the roller coaster goes, the first two groups [1, 4] will ride, leaving an empty seat (the group of 2 won't fit, and the group of 1 can't go ahead of them). Then they'll go to the back of the queue, which now looks like 2, 1, 1, 4. The second time, the coaster will hold 4 people: [2, 1, 1]. Now the queue looks like 4, 2, 1, 1. The third time, it will hold 6 people: [4, 2]. Now the queue looks like [1, 1, 4, 2]. Finally, it will hold 6 people: [1, 1, 4]. The roller coaster has made a total of 21 Euros!

Entrada

The first line of the input gives the number of test cases, **T**. **T** test cases follow, with each test case consisting of two lines. The first line contains three space-separated integers: **R**, **k** and **N**. The second line contains **N** space-separated integers $\mathbf{g_i}$, each of which is the size of a group that wants to ride. $\mathbf{g_0}$ is the size of the first group, $\mathbf{g_1}$ is the size of the second group, etc.

Salida

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is the number of Euros made by the roller coaster.

Entrada	Salida
3 4 6 4 1 4 2 1 100 10 1 1 5 5 10 2 4 2 3 4 2 1 2 1 3	Case #1: 21 Case #2: 100 Case #3: 20