Competencia de Programación

TecnoMate

Nivel “LIBRES”

31 de Octubre de 2014

Problema 1: CARTONES CREDECRECIENTES (11732. CREDECRE)

Para una variante de la tómbola, se quiere confeccionar e imprimir cartones del siguiente tipo:

T1				T2				T3		
1	2	3	.	1	2	4	.	1	2	5
6	5	4	.	7	6	5	.	8	7	6
7	8	9	.	8	9	10	.	9	10	11

En la matriz, se deben ubicar los números en 1..N, los cuales pueden aparecer una vez como máximo -en cada cartón-. Por ejemplo, T1, es un cartón válido cuando N=9.

T2 es un ejemplo de un cartón válido cuando N=10, y T3 es un ejemplo de un cartón válido cuando N=11.

Los tableros válidos cumplen con las siguientes condiciones:

- Los elemenos de la fila 1 tienen que estar ordenados en forma creciente.
- Los elementos de la fila 2 tienen que estar ordenados en forma decreciente.
- Los elementos de la fila 3 tienen que estar ordenados en forma creciente.
- El menor de los elementos de la fila 2 tiene que ser mayor que el mayor de la fila 1
- El menor de los elementos de la fila 3 tiene que ser mayor que el mayor de la fila 2

Los cartones siempre son de 3x3, pero el número N puede variar. Se está organizando un evento multitudinario y se quiere conocer la cantidad de cartones válidos para diferentes valores de N. Ud. debe codificar un programa que lea valores para N, ingresados por teclado, e informe la cantidad de cartones válidos diferentes que se pueden confeccionar para dicho valor de N.
La entrada de datos termina cuando ingresa N=0, los valores de N están en el rango 0..24

Ejemplo
Entrada

```
4
9
11
10
24
0
```

Salida

```
0
1
55
10
```

1307504

Problema 2: Bomba (15991. AI_BOM)

Estas encerrado dentro de un castillo que solo tiene una puerta de salida que esta cerrada por dentro y por fuera. Existe una bomba dentro y debes escapar antes de que esta explote pero antes debes encontrar la llave de la única salida.
Solo puedes moverte hacia arriba, abajo, izquierda y derecha, por el poco tiempo que tienes, ya que no sabes cuando explotara la bomba, cual es la mínima cantidad de movimientos que necesitas para salir?

Entrada

La entrada consiste en 10 lineas, cada linea contiene 10 caracteres ('o', 'x', 'm', 'k' y 'e') que representan el castillo dividido en celdas. Donde 'o' representa un lugar en donde puedes moverte, 'x' son las paredes y no por lo tanto no puedes moverte allí, tu posición actual es representada por la letra 'm', la llave esta representada por la letra 'k' y finalmente la salida es representada por la letra 'e'. Se garantiza que cada castillo tendrá una llave y salida y al menos un camino valido.

Salida

Imprimir un simple numero entero, la cantidad mínima de movimientos que necesitas para salir.

Ejemplo

Entrada:	**Salida:**
xxxxxxxxxx xmooxxxxxx xoooooooox xxooxxxxxx xookxxxxxx xoooxxxxxx xxxooxxxxx xxxoooxxxx xxxooooooe xxxxxxxxxx	15
xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx xoooooooox xooomoooox xoooooooox xxxokoooxx xxxooooooe xxxoooooxx xxxxxxxxxx	8

Problema 3: Caperucita Roja (16277. TAP2013C)

Erase una vez una niña muy alegre a la que llamaban Caperucita Roja porque siempre vestía una caperuza de color rojo. Caperucita disfrutaba mucho de sus paseos por el bosque, durante los cuales recolectaba frutos en su canastita para llevárselos a su abuelita, quien era conocida por preparar las tartas más deliciosas de toda la región. De lo que definitivamente no disfrutaba Caperucita era de los peligros del bosque, y en particular del malvado lobo que siempre estaba hambriento y al acecho.

Un día Caperucita decide ir desde su casa hasta la de su abuelita, recolectando frutos del bosque en el camino y tratando de hacer el paseo de la manera más segura posible. La casa de Caperucita está en un claro del extremo oeste del bosque, la casa de su abuelita en un claro del extremo este, y dentro del bosque entre ambas hay algunos otros claros con arboles frutales. El bosque es muy espeso, por lo que la única forma de atravesarlo es utilizando senderos entre los distintos claros, los cuales por suerte Caperucita conoce muy bien. Para no perderse Caperucita se mueve siempre por senderos que la llevan a un punto estrictamente al este del punto en el que se encuentra. Para no ser atrapada por el lobo Caperucita encuentra imprescindible evitar una emboscada, y para ello siempre tiene en cuenta la cantidad de caminos distintos que la llevan desde su posición actual hasta la casa de su abuelita.

Un camino en el bosque es una sucesión de claros ordenados de oeste a este, de manera que cada claro está conectado con el siguiente por un sendero. Un camino hasta la casa de la abuelita es simplemente un camino cuyo último claro contiene dicha casa. Para cada claro, su *nivel de alternativas* es la cantidad de caminos que van desde él hasta la casa de la abuelita. Para cada camino, su *nivel de alternativas* es la suma de los niveles de alternativas de cada claro que compone ese camino. Para no ser capturada por el lobo, Caperucita desea encontrar el camino con máximo nivel de alternativas que empieza por su casa y termina en la de su abuelita. Como miembros de la sociedad Ayuda a Caperucita a Moverse (ACM), están llamados a ayudarla a encontrar ese valor máximo, y cuando lo hayan hecho, colorín, colorado, este problema habrá terminado.

Entrada

La primera línea contiene dos enteros N y S que indican respectivamente la cantidad de claros y de senderos en el bosque ($3 \leq N \leq 3 \times 10^4$ y $2 \leq S \leq 10^5$). Los claros son identificados por enteros diferentes entre 1 y N y están ordenados de oeste a este, de modo que si $1 \leq i < j \leq N$ el claro i está más al oeste que el claro j. La casa de Caperucita está en el claro 1, y la casa de la abuelita está en el claro N. Cada una de las S líneas siguientes describe un sendero utilizando dos enteros I y J que representan que hay un sendero entre entre el claro I y el claro J ($1 \leq I < J \leq N$). Hay al menos un camino desde la casa de Caperucita hasta la casa de la abuelita, y el máximo nivel de alternativas dentro del conjunto de tales caminos es menor o igual que 10^{18}.

Salida

Imprimir en la salida una línea conteniendo un entero que representa el máximo nivel de alternativas de un camino desde la casa de Caperucita hasta la de su abuelita.

Ejemplo

Entrada:	**Output:**

3 2 1 2 2 3	3
4 6 1 2 2 3 3 4 1 2 2 3 3 4	15
9 9 1 3 2 3 3 4 4 5 1 5 3 4 3 9 7 8 4 9	8

Problema 4: Guerra (16281. TAP2013G)

La guerra, evento solamente digno de aparecer en la literatura, películas o quizás problemas de competencias de programación, llegó al imperio de Nlogonia, que se enfrenta al vecino imperio de Quadradonia.

Los protocolos de guerra pautados por ambos imperios indican que la guerra será desarrollada en sucesivas batallas, y que en cada una de estas batallas se enfrentará un soldado distinto de cada imperio, de modo que cada soldado participará en exactamente una batalla. El imperio que gane la mayor cantidad de batallas ganará la guerra.

Cada imperio posee un ejército conformado por S soldados, y cada soldado posee cierta habilidad para el combate. En cada batalla entre dos soldados, aquel con mayor habilidad para el combate gana la batalla. Si ambos soldados poseen la misma habilidad, la batalla resulta un empate y técnicamente ningún bando se adjudica la victoria. Los espías de Nlogonia interceptaron información secreta sobre la habilidad de cada soldado de Quadradonia, y la reina de Nlogonia requiere la ayuda de ustedes para saber la cantidad máxima de batallas que puede ganar en la guerra si envía a sus soldados en el orden adecuado.

Input

La primera línea contiene un entero S que indica la cantidad de soldados de cada imperio ($1 \leq S \leq 10^5$). La segunda línea contiene S enteros Q_i que representan las habilidades de los distintos soldados de Quadradonia, en el orden en que se sucederán las batallas ($1 \leq Q_i \leq 10^9$ para i = 1, 2, . . . , S). La tercera línea contiene S enteros N_i que representan las habilidades de los distintos soldados de Nlogonia, en un orden cualquiera ($1 \leq N_i \leq 10^9$ para i = 1, 2, . . . , S).

Output

Imprimir en la salida una línea conteniendo un entero que representa la cantidad máxima de batallas que puede ganar Nlogonia en la guerra.

Ejemplo:

Entrada:	**Salida:**
3 2 1 1000000000 1 1 2	1

Problema 5: El Juego de las Fracciones (16416. AI_JUE)

Alice y Bob se divierten con el siguiente juego, cada uno escribe en un papel una acordada cantidad de fracciones en tan solo 5 segundos, El ganador es la persona que escribió mas fracciones irreducibles. Por ejemplo si la cantidad acordada de fracciones es 3 y Alice escribió: 1/12 , 7/12 y 3/12 , Bob escribió: 3/7 , 2/8 y 10/5. Gana Alice porque tiene dos fracciones irreducibles 1/12 y 7/12 y Bob solo tiene una 3/7.
Ayuda a Alice y Bob a decidir quien es el ganador.

Input

La entrada contiene varios casos de prueba cada uno descrito en tres lineas. La primera linea contiene un numero entero N indicando el numero de fracciones acordado entre Alice y Bob escritos en un papel ($1 \leq N \leq 100$). La segunda linea contiene N fracciones X_i separados por un espacio cada uno con un numerados n_i y denominador d_i, representando las fracciones que Alice escribió n_i/d_i ($1 \leq n_i, d_i \leq 100$ para $1 \leq i \leq N$). La tercera linea contiene N fracciones Y_i separados por un simple espacio, representando las fracciones que Bob escribió ($1 \leq Y_i \leq 100$ para $1 \leq i \leq N$). La entrada termina con $N = 0$.

Output

Para cada caso de prueba imprimir una sola linea "Alice" si Alice gana el juego, o la cadena "Bob" si Bob gana el jugo, en el caso de empate imprimir "=".

Ejemplos

<table>
<tr><th>Input:</th><th>Output:</th></tr>
<tr><td><pre>3
1/12 7/12 3/12
3/7 2/8 10/5
5
2/4 6/3 8/2 1/3 15/3
3/5 9/11 4/2 56/4 33/3
1
7/2
4/5
0</pre></td><td><pre>Alice
Bob
=</pre></td></tr>
</table>

Problema 6: Al azar (21168. TAP2014A)

Los juegos de cartas son muchos y muy variados, y su origen se remonta a tiempos ancestrales. A veces puede resultar sorprendente que sigan siendo capaces de proveernos entretenimiento después de siglos de ser jugados con las mismas reglas, pero entonces debemos comprender que cada partida es esencialmente distinta de todas las demás que se han jugado en la historia de la humanidad, dada la gran cantidad de posibles formas de ordenar las cartas antes de iniciarla. En efecto, pocos juegos resultan entretenidos si utilizamos las cartas siempre en el mismo orden, o si existe una correlación entre cartas sucesivas que nos permita predecir el orden en el que se encuentran. Esta es la razón por la cual se suele mezclar las cartas antes de empezar cada juego, y por esto mismo les pedimos ahora que hagan un programa para controlar que una secuencia de cartas ha sido bien mezclada.

Para simplificar el problema, vamos a concentrarnos solamente en las barajas de cartas españolas, que consisten en 48 cartas distintas. Cada carta está identificada por un valor, que es un número del 1 al 12, y por un palo, que puede ser “bastos”, “copas” ,“espadas” u
“oros”. Ahora bien, como no queremos simplificar excesivamente su tarea, vamos a tener en cuenta que no todos los juegos utilizan las 48 cartas de la baraja. Dada una secuencia de N cartas, decimos que está bien mezclada si no hay en ella dos cartas sucesivas que comparten el mismo valor o el mismo palo. Caso contrario, decimos que ha sido mal mezclada. ¿Pueden ayudarnos a decidir si una secuencia está bien mezclada?

Input

La primer línea contiene un entero T, el número de casos de prueba ($2 \leq T \leq 48$). Le siguen T casos de prueba.

La primera línea de cada caso de prueba contiene un entero N, que representa la cantidad de cartas que se utilizan en el juego que vamos a considerar ($2 \leq N \leq 48$). Cada una de las siguientes N líneas contiene la descripción de una carta de la secuencia que queremos analizar, dada por un entero V que representa su valor ($1 \leq V \leq 12$) y un carácter P que representa su palo: “b” para bastos; “c” para copas; “e” para espadas y “o” para oros. Todas las cartas dadas son distintas, y se dan en la entrada en el mismo orden en el que aparecen en la secuencia.

Output

Imprimir en la salida una línea conteniendo un carácter que representa si la secuencia de cartas dada en la entrada ha sido bien mezclada o no. El carácter debe ser “B” en caso de que esté bien mezclada y “M” en caso de que esté mal mezclada.

Input:	**Output:**

`4`
`4`
`1 b`
	`B`
	`M`
	`M`

2 c 3 e 4 o 3 1 b 2 b 3 c 3 1 b 1 c 2 e 32 5 c 2 b 4 e 3 o 12 b 1 c 7 e 6 c 12 e 4 o 1 b 6 o 3 e 12 o 11 e 12 c 5 o 10 b 9 o 3 c 4 b 11 c 8 e 9 c 1 e 4 c 8 b 2 o 6 b 9 e 7 b 5 e	B

Problema 7: Base-3 balanceada (21169. TAP2014B)

A lo largo de la historia se han desarrollado sistemas de numeración muy diversos. Algunos, como el de números romanos, se han abandonado casi por completo en la actualidad por ser poco convenientes. Otros sistemas aún más exóticos son utilizados sólo para ciertas aplicaciones, como por ejemplo el sistema factorádico en el contexto de la numeración de permutaciones. En este problema vamos a considerar un sistema de numeración conocido como *base-3 balanceada*, que surge naturalmente en el análisis de diversos problemas matemáticos relacionados con las balanzas de platillos.

El sistema de base-3 balanceada es similar al sistema decimal o base-10 al que estamos acostumbrados, que es un sistema *posicional*. Los sistemas posicionales tienen *dígitos* cuyo orden relativo determina qué potencia de la *base* los acompaña. Por ejemplo, en base-10 tenemos

$$123 = \mathbf{1} \times 10^2 + \mathbf{2} \times 10^1 + \mathbf{3} \times 10^0.$$

En los sistemas posicionales estándar, los dígitos permitidos son todos los números del 0 al $B - 1$, donde B es la base del sistema utilizado. Entonces, el número 123 en base-10 queda escrito en base-3 estándar como "11120", dado que

$$\mathbf{1} \times 3^4 + \mathbf{1} \times 3^3 + \mathbf{1} \times 3^2 + \mathbf{2} \times 3^1 + \mathbf{0} \times 3^0 = 123.$$

La base-3 balanceada difiere de la base-3 estándar solamente en el hecho de que los dígitos permitidos son el 0, el 1 y el −1, que vamos a escribir como "**0**", "**+**" y "**-**" respectivamente. De este modo, el número 123 en base-10 queda escrito como "**+----0**" en base-3 balanceada, porque

$$\mathbf{1} \times 3^5 + (\mathbf{-1}) \times 3^4 + (\mathbf{-1}) \times 3^3 + (\mathbf{-1}) \times 3^2 + (\mathbf{-1}) \times 3^1 + \mathbf{0} \times 3^0 = 123.$$

Como la conversión de números en base-10 a base-3 balanceada es un proceso mecánico y algo tedioso, requerimos un programa que lo realice por nosotros. ¿Pueden ayudarnos?

Entrada:

La primera línea contiene un número entero T que representa el número de casos de prueba ($1 \leq T \leq 200$). A continuación siguen T casos de prueba.

Cada caso de prueba consiste en una única línea con un entero positivo N, el número en base-10 que deseamos escribir en base-3 balanceada ($1 \leq N \leq 1000$).

Salida:

Para cada caso de prueba, imprimir en la salida una línea conteniendo una cadena formada únicamente por los caracteres "0", "+" y "-" y que no empiece con "0", que representa los dígitos del número N en base-3 balanceada. La restricción de no comenzar la secuencia con "0" asegura que la representación es única.

Entrada de ejemplo	Salida para la entrada de ejemplo
2 123 729	+----0 +000000

Problema 8: Batuke (21593. BATUKE)

Batuke es un perro con comportamiento repetitivo que tiene una rutina particular cuando sale a dar un paseo por su barrio. El barrio también es particular; una matriz cuadrada de NxN, que Batuke sabe recorrer con una rutina en espiral: cuando comienza hace: 1 casilla a derecha, una abajo, luego dos a izquierda, dos arriba, luego 3 a derecha, 3 abajo, luego 4 ….
A Batuke, Lucas (su dueño) lo lleva en auto a la casilla de partida, y Batuke siempre sigue su rutina, hasta que recorre todas las casillas de su barrio.
Por ejemplo si su barrio es de tamaño N=4, el mismo se numera de la siguiente forma:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

y el recorrido de Batuke, partiendo de la casilla 1,1 (la casilla superior izquierda es la 0,0), es:
6,7,11,10,9,5,1,2,3,4,8,12,16,15,14,13

En este caso su rutina lo hace caminar 16 casillas.

Pero Batuke no distingue los límites del barrio y nunca se sale de su rutina (aunque tenga que caminar más para recorrer todas las casillas de su barrio). Si la casilla de partida es la 2,2; el recorrido en espiral (sólo se muestran las casillas del barrio) es el siguiente:
11,12,16,15,14,10,6,7,8,13,9,5,1,2,3,4. En este caso su rutina lo hace caminar 24 casillas.
Lucas sabe que estudiás ISI, y te pide que dado un barrio de N filas x columnas, y una casilla donde él deja a Batuke, le informés el recorrido en espiral, y el total de cuadras que caminó Batuke.

Entrada:

La entrada consiste de tres números enteros: N, F, C. N, 2<N<=10 indica la cantidad de filas que tiene el barrio (con celdas numeradas de 1 a NxN, por filas y de izquierda a derecha). F y C, indican la fila y columna de la casilla donde Batuke comienza su recorrido.

Salida:

La salida consiste de una línea conteniendo una lista con las casillas del barrio ordenadas según el recorrido, y separadas por un espacio en blanco. La línea siguiente contiene un número entero que muestra la cantidad de cuadras que caminó Batuke.

Input	**Output**
4 1 1	6 7 11 10 9 5 1 2 3 4 8 12 16 15 14 13 16
4 3 3	16 15 11 12 14 10 6 7 8 13 9 5 1 2 3 4 46