

Maratón de Programación - 2014 - Semifinal

CATEGORÍA Grado 11

30/10/2014

Politécnico Colombiano Jaime Isaza Cadavid



**Reglas de la competencia**

**Restricciones:**

1. Los equipos participantes no pueden utilizar ningún tipo de material digital para la realización de los ejercicios.
2. Ninguno de los miembros del equipo puede hablar con los competidores de los otros equipos participantes.
3. Los equipos no pueden utilizar a internet (Navegar) en ningún momento.
4. Cada uno de los integrantes del equipo debe apagar sus equipos móviles (Celulares, Palm’s, portátiles u otros similares).
5. Los miembros de equipo solo pueden usar el computador asignado un lápiz o lapicero y hojas en blanco.

**Método de trabajo:**

1. A cada equipo se le asignará de manera aleatoria uno de los computadores.
2. Los participantes recibirán instrucciones precisas de la forma de trabajo y las reglas del concurso.
3. Cada uno de los equipos tiene aproximadamente 4 horas para completar la prueba.
4. Los ejercicios pueden ser resueltos en diferente orden.
5. Cada ejercicio tiene igual puntuación.
6. Los equipos se clasificarán por tiempo, es decir se suman los tiempos desde el inicio de la prueba hasta la solución de cada ejercicio y se organizan los tiempos de menor a mayor.
7. Si alguno de los equipos termina correctamente (y en el menor tiempo) la solución de todos los ejercicios antes de las 4 horas planeadas se convierte inmediatamente en el ganador.

Los jurados y docentes tienen plena libertad de sancionar con la descalificación cualquier conducta irregular en el desarrollo de la prueba.



**ÉXITOS!!!**

**RETO 1**

**DÓNDE ESTÁ LA PALABRA?**

**Nombre del archivo fuente:** Donde.c, Donde.cpp, o Donde.java

Dada una matriz de m por n letras y una lista de palabras, encontrar la ubicación en la matriz, en la cual se encuentran las palabras.

Una palabra en la matriz corresponde a una línea recta ininterrumpida de letras, que puede estar compuesta por letras en mayúsculas o minúsculas. La coincidencia se puede dar en cualquiera de las 8 direcciones horizontales, verticales o diagonales.

**Datos de entrada** (entrada estándar):

La entrada comienza con un único entero positivo en una línea, (el cual indica el número de casos a realizar).

Cada caso comienza con una pareja de enteros, m seguido de n, en una única línea, donde 1 ≤ m; n ≤ 50, en notación decimal. Las siguientes m líneas contendrán n letras cada una, representando la tabla de letras en la que se deben buscar las palabras. Las letras de la tabla pueden estar en mayúsculas o minúsculas. A continuación de la tabla de letras, aparecerá una línea con otro entero k (1 ≤ k≥ 20). Las siguientes k líneas de la entrada contendrán la lista de palabras que se deben buscar, una por línea. Estas palabras pueden contener únicamente letras mayúsculas y minúsculas, y no aparecerá ningún espacio, signo de puntuación o carácter no alfabético.

**Datos de salida** (salida estándar):

Para cada palabra de cada caso de prueba, se debe mostrar una pareja de enteros que representará su ubicación en la tabla correspondiente. Los enteros deben estar separados por un único espacio.

El primer entero corresponde a la línea de la tabla donde aparece la primera letra de la palabra en cuestión (1 representa la línea superior de la tabla, y m la inferior). El segundo entero es la columna de la tabla en la que aparece la primera letra de la palabra (1 representa la columna del extremo izquierdo, y n la del extremo derecho). Si una palabra aparece más de una vez en la tabla, se debe mostrar la que esté situada más arriba (es decir, la que tenga su primera letra en el lugar más cercano a la parte superior de la tabla). En caso de que ambas apariciones se encuentren a la misma altura, se considerará la que esté más a la izquierda. Todas las palabras aparecerán, al menos, una vez en la tabla.

La salida de dos casos de prueba consecutivos debe estar separada por una línea en blanco**.**

**Ejemplo de entrada**

1

8 11

abcDEFGhigg

hEbkWalDork

FtyAwaldORm

FtsimrLqsrc

byoArBeDeyv

Klcbqwikomk

strEBGadhrb

yUiqlxcnBjf

4

Waldorf

Bambi

Betty

Dagbert

**Ejemplo de salida**

2 5

2 3

1 2

78

**RETO 2**

**ESPEJO**

**Nombre del archivo fuente:** espejo.c, espejo.cpp, o espejo.java

Supongamos que se nos da una matriz **b** de la medida **x** \* **y**, vamos a determinar el funcionamiento para crear el espejo de la matriz **b**. La matriz resultante de la creación del espejo de la matriz **b** es una matriz **c** de **2x** \* **y**, que tiene las siguientes propiedades:

* La mitad superior de la matriz **c** (filas con números del 1 al **x** ) coincide exactamente con **b**;
* La mitad inferior de la matriz **c** (filas con números de **x** + 1 a **2x** ) es simétrica a la superior;
* En este caso la línea de simetría será la línea que separa las dos mitades (la línea que va en el medio, entre las filas de **x** y **x + 1** ) .

Si se tiene la matriz ***a*** de **m \* n** y se quiere encontrar la matriz **b**, que se puede transformar en la matriz ***a***, si vamos a realizar sobre ella varios (posiblemente cero) reflejos. ¿El mínimo número de filas que puede contener tal matriz es?

ENTRADA

* La primera línea contiene dos enteros, n y m ( 1 ≤ n , m ≤ 100 ).
* Cada una de las siguientes n líneas contiene m enteros - Los elementos de la matriz ***a.***
* La línea de orden ***i*** contiene enteros ***ai*1, *ai*2, ..., *aim*** (0 ≤ ***aij***≤ 1) — La *i*- esima fila de la matriz ***a***.

SALIDA

* En la única línea imprimir la respuesta al problema - el número mínimo de filas de la matriz b.

EJEMPLOS

ENTRADA

4 3  
0 0 1  
1 1 0  
1 1 0  
0 0 1

SALIDA

2

ENTRADA

3 3  
0 0 0  
0 0 0  
0 0 0

SALIDA

3

ENTRADA

8 1  
0  
1  
1  
0  
0  
1  
1  
0

SALIDA

2

**RETO 3**

**LABERINTO**

**Nombre del archivo fuente:** laberinto.c, laberinto.cpp, o laberinto.java

Un día, El pequeño Suertudo se encontró en un laberinto que consistía en ( n + 1 ) habitaciones, numeradas de 1 a (n + 1 ). Inicialmente, Suertudo se encuentra en la primera habitación y para salir del laberinto necesita llegar a la (n + 1 ) - ésima habitación.

El laberinto está organizado de la siguiente manera. Todas las habitaciones del laberinto tienen dos puertas en un solo sentido. Vamos a considerar el número de habitación *i* (1 ≤ *i* ≤ *n*), alguien puede utilizar la primera puerta para pasar a la habitación número ( *i* + 1 ), también alguien puede utilizar la segundapuerta para pasar a la habitación número *pi*, donde 1 ≤ *pi* ≤ *i*.

Para no perderse, Suertudo decidió actuar de la siguiente manera:

* Cada vez que Suertudo entra en un cuarto, pinta una cruz en su techo. Inicialmente, Suertudo pinta una cruz en el techo de la habitación 1.
* Vamos a suponer que Suertudo está en habitación *i* y ya ha pintado una cruz en su techo. Entonces, si el techo contiene un número impar de cruces, Suertudo utiliza la segunda puerta (que conduce a la sala de *pi*), de lo contrario utiliza la primera puerta.

Ayuda a Suertudo a determinar el número de veces que tiene que utilizar las puertas para llegar a la habitación (n + 1 ) en la salida.

ENTRADA

* La primera línea contiene un entero ***n*** ( 1 ≤ n ≤ 103 ) - el número de habitaciones.
* La segunda línea contiene n enteros *pi* (1 ≤ *pi* ≤ *i*.). Cada *pi* denota el número de la habitación a la que alguien puede llegar si va a usar el segundo portal en la sala *i*.

SALIDA

* Imprimir un único número: el número de puertas que tiene que pasar Suertudo para salir del laberinto. Ya que el número puede ser muy grande imprimirlo como módulo de 1000000007 (109 + 7).

EJEMPLOS:

ENTRADA

2  
1 2

SALIDA

4

ENTRADA

4  
1 1 2 3

SALIDA

20

ENTRADA

20

1 2 2 2 2 1 4 7 8 6 5 3 5 3 8 11 5 10 16 10

SALIDA

433410

**RETO 4**

**PRISON TRANSFER**

**Nombre del archivo fuente:** prision.c, prisioncpp, o prision.java

The prison of your city has *n* prisoners. As the prison can't accommodate all of them, the city mayor has decided to transfer *c* of the prisoners to a prison located in another city.

For this reason, he made the *n* prisoners to stand in a line, with a number written on their chests. The number is the severity of the crime he/she has committed. The greater the number, the more severe his/her crime was.

Then, the mayor told you to choose the *c* prisoners, who will be transferred to the other prison. He also imposed two conditions. They are,

* The chosen *c* prisoners has to form a contiguous segment of prisoners.
* Any of the chosen prisoner's crime level should not be greater then *t*. Because, that will make the prisoner a severe criminal and the mayor doesn't want to take the risk of his running away during the transfer.

Find the number of ways you can choose the *c* prisoners.

**Input**

The first line of input will contain three space separated integers *n* (1 ≤ *n* ≤ 2·105), *t* (0 ≤ *t* ≤ 109) and *c* (1 ≤ *c* ≤ *n*). The next line will contain *n* space separated integers, the *ith* integer is the severity *ith* prisoner's crime. The value of crime severities will be non-negative and will not exceed 109.

**Output**

Print a single integer — the number of ways you can choose the *c* prisoners.

Sample test

Input

4 3 3  
2 3 1 1

Output

2

Input

1 1 1  
2

Output

0

Input

11 4 2  
2 2 0 7 3 2 2 4 9 1 4

Output

6