

Nivel “Secundaria”
Problemas

14 de Octubre de 2016

Problema 1: Detectando colisiones

La detección de colisiones es una de las operaciones más comunes (e importantes) en los juegos electrónicos. El objetivo básicamente es verificar que dos objetos chocan entre sí, es decir, si la intersección entre ellos es diferente de vacío. Esto se puede utilizar para saber si dos barcos chocaron, si un monstruo chocó contra un muro, si un personaje recogió un elemento, etc.

Para facilitar las cosas, muchas veces los objetos se aproximan mediante formas geométricas simples (esferas, triángulos, adoquines, etc). En este problema, los objetos se aproximan mediante rectángulos en un plano 2D.

Escribir un programa que, dado dos rectángulos, determine si se cruzan o no.

Entrada

La entrada contiene un único conjunto de pruebas que deben ser leídos desde el dispositivo de entrada estándar (típicamente un teclado). Cada caso de prueba contiene dos líneas. Cada línea contiene cuatro números enteros (x0, y0, x1, y1, siendo $0 \leq x0 < x1 \leq 1.000.000$ y $0 \leq y0 < y1 \leq 1.000.000$) separados por un espacio en blanco que representa un rectángulo. Los lados del rectángulo son siempre paralelos a los ejes x e y.

Salida

Su programa debe imprimir en la salida estándar, una línea para cada caso de prueba que contenga el número 0 (cero) si no hay intersección ó el número 1 (uno) si la hubiere.

Ejemplos

Entrada	Salida
0 0 1 1 0 0 1 1	1
0 0 2 2 1 1 3 3	1
0 0 1 1 2 2 3 3	0

Problema 2: Funciones

En la última clase de matemáticas, Rafael, Beto y Carlos aprendieron un nuevo conjunto de funciones matemáticas. Cada uno de ellos seleccionó una función específica a fin de competir y determinar cuál de ellas tiene una salida mayor.
La función elegida por Rafael es $r(x, y) = (3x)^2 + y^2$.
La función elegida por Beto es $b(x, y) = 2(x^2) + (5y)^2$.
La función elegida por Carlos es $c(x, y) = -100x + y^3$.
Dados los valores de x e y, determinar quién eligió la función que tiene el mayor valor de salida.

Entrada

La primera línea contiene un entero N que determina la cantidad de casos de prueba. Cada caso de prueba consiste en dos valores enteros x e y ($1 \leq x, y \leq 100$), que indican las variables de entrada para cada una de las funciones.

Salida

Para cada caso de prueba imprimir una línea que contenga una única oración que indique quien ganó el concurso (asumiendo que no se darán empates). Por ejemplo, si ganó Rafael, se deberá imprimir “Rafael ganhou”.

Ejemplo

Entrada	Salida
6 5 3 2 30 2 100 30 20 15 5 30 2	Beto ganhou Carlos ganhou Carlos ganhou Beto ganhou Rafael ganhou Rafael ganhou

Problema 3: Suma Natural

Un número natural es un entero no-negativo (0, 1, 2, 3, 4, 5,...). Tu tarea en este problema es calcular la suma de los números naturales que están en un determinado intervalo [**A**, **B**] inclusive.

Por ejemplo, la suma de los números naturales en el intervalo [2, 5] es 14 = (2 + 3 + 4 + 5).

Entrada

La entrada contiene un caso de prueba con dos enteros **A** y **B** ($1 \leq \mathbf{A} \leq \mathbf{B} \leq 10^9$), que representan los límites inferior y superior respectivamente.

Salida

Para cada caso de prueba, la salida consiste de una línea que contiene la suma de los números naturales del intervalo.

Ejemplo

Entrada	Salida
2 5	14
10 20	165
1 1000	500500

Problema 4: OBI URI

Mariana ha creado un ejercicio para sus hermanas, Paula y Marta: ella distribuye un texto y a ambas les pide que lo corrijan, sabiendo que solamente las palabras OBI y URI pueden estar escritas incorrectamente, y el error solamente estará en la última letra.

Tu tarea aquí es automatizar este proceso, creando un programa para realizar la corrección de los textos distribuidos por Mariana que verifiquen las correcciones de sus hermanas sin mucho trabajo.

Observá que si la secuencia "OB" o "UR" son partes de una palabra mayor (por ejemplo, "OBOS"), la misma no deberá ser alterada.

Entrada

La entrada contiene dos líneas. La primera, contiene un número entero N (1 < N < 10000) que indica la cantidad de palabras del texto. La segunda línea, contiene el texto que debe ser corregido.

Salida

Tu programa debe imprimir el texto según las modificaciones explicadas anteriormente.

Ejemplo

Entrada	Salida
2 OBO URU	OBI URI
3 EURO AVOID OBITS	EURO AVOID OBITS
10 URA URO URI URU UROS IBO OBA OBAS OBES OBE	URI URI URI URI UROS IBO OBI OBAS OBES OBI

Problema 5: Juego de estudiantes

A Juilherme y Jogerio les gustan los juegos matemáticos. Por este motivo, Juilherme ha creado un nuevo juego que consta de los siguientes pasos:
1) Juilherme elige un valor entero N y Jogerio elige un valor entero M.
2) Juilherme y Jogerio deben encontrar dos números primos P1 y P2 que sean los enteros más cercanos posibles a N y M, respectivamente. Además, P1 debe ser igual o menor que N y P2 debe ser igual o menor que M.
3) La respuesta final del juego es el valor resultante de multiplicar P1 y P2. Quien encuentre primero este valor será el ganador.
Ambos jugadores tratarán de encontrar la respuesta lo más rápido posible y, alguna que otra vez, pueden equivocarse. Por este motivo, necesitan un algoritmo que determine la respuesta final a fin de comparar la respuesta que ellos han determinado individualmente con la respuesta correcta.
Utilizando la información de este juego, crear un nuevo programa que dados los valores de N y M, imprima el valor resultante del juego.

Entrada

La entrada será una única línea con los valores de N y M (2 <= N, M <= 1000).

Salida

La salida del programa será una única línea con la respuesta final del juego.

Ejemplos:

Entrada	Salida
10 15	91
50 100	4559

Problema 6: Colección de Pomekons



Desde que está oficialmente lanzada pomekon en Brasil, Gabriel está intentando realizar su gran sueño: Ser un maestro pokémon. Su meta es conquistar los 151 pomekon disponibles. Él logró la captura de muchos monstruos, pero en tu ciudad aparecen muchos pomekon repetidos, haciendo que se capture varias veces el mismo pomekon.

Como siempre vé tu mochila llena de libros de programación, Dabriel te pide que hagas un programa que le informe cuantos pomekon faltan para completar la colección.

Entrada

La primera línea del caso de prueba consiste en un entero **N** ($1 \leq \mathbf{N} \leq 10^3$), representando la cantidad de pomekons que Dabriel ya ha capturado.

Las siguientes **N** líneas consisten en cadenas de caracteres **S** ($1 \leq \mathbf{|S|} \leq 10^3$) representando el nombre de cada pomekon. El nombre de cada pomekon consiste solamente de letras mayúsculas y minúsculas.

Salida

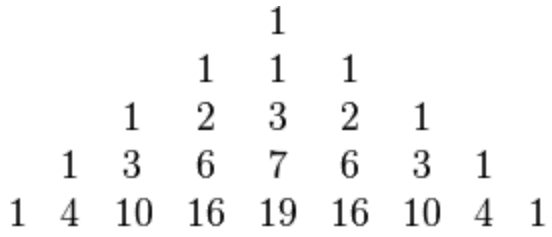
Debe imprimir: "Falta(m) **X** pomekon(s).", Donde **X** es la cantidad de pomekons no capturados.

Ejemplos:

Entrada	**Salida**
7 Charmander Caterpie Pidgeot Rattata Zubat Zubat Zubat	Falta(m) 146 pomekon(s).
8 Zubat Zubat Zubat Zubat Zubat Zubat Zubat Zubat	Falta(m) 150 pomekon(s).

Problema 7: Triángulo Trinomial

El triángulo trinomial es un triángulo numérico de coeficientes trinomiales. Este triángulo puede ser obtenido a partir de una fila que contiene un único "1", la siguiente fila contiene tres "1" y cada elemento de las filas siguientes se calcula sumando los elementos que están arriba a la izquierda, inmediatamente arriba y arriba a la derecha:



La primera fila del triángulo trinomial se numera con cero, la segunda fila es la número 1 y así sucesivamente.

Tu tarea es, dado un número de fila *R*, escribir un programa que muestre la suma de sus elementos. Por ejemplo, la suma de los elementos de la fila 2 es 9 = 1 + 2 + 3 + 2 + 1.

Entrada

La entrada es el número de fila *R* ($0 \leq R \leq 20$).

Salida

La salida es la suma de todos los elementos de la fila *R*. No olvide el caracter de fin de línea luego de exhibir la suma.

Ejemplos:

Entrada	Salida
0	1
1	3
2	9

Problema 8: Campo de Lombrices

Las lombrices de tierra son muy importantes tanto para la agricultura como para la materia prima en la producción de alimentos para animales. La Organización para la Bioingeniería de las lombrices de tierra (OBM) es una organización no gubernamental que promueve el aumento de la producción, el uso y la exportación de las lombrices.
Una de las actividades promovidas por la OBM es el mantenimiento de una granja experimental para la investigación de nuevas tecnologías para crear lombrices. En la granja, la superficie dedicada a la investigación es de forma rectangular, y se encuentra dividida en celdas cuadradas del mismo tamaño. En cada celda se crea sólo un tipo de lombriz. Las células se utilizan para probar los efectos sobre la producción de lombrices de las variaciones de especies de lombrices, tipo de suelo, el fertilizante, la humedad, etc. Los investigadores de la OBM mantienen una vigilancia constante del desarrollo de lombrices en cada celda, y tienen una estimación muy precisa de la productividad de cada una de ellas.

81	28	240	10
40	10	100	240
20	180	110	35

Un investigador de la OBM inventó y construyó una máquina cosechadora de lombrices y decidió probarla en esta granja. La máquina tiene la anchura de una celda, y trabaja desde el centro del suelo recogiendo todos las lombrices en la misma, separándolas, limpiándolas y envasádolas para su comercialización. Es decir, esta máquina elimina uno de los pasos de mano de obra intensiva más costosos del proceso de producción de lombrices. Sin embargo, como la máquina aún está todavía en desarrollo tiene una limitación: sólo puede moverse en línea recta.

Teniendo en cuenta esta restricción, se decidió probar la máquina a fin de obtener el mayor número posible de lombrices en una sola pasada (recta, de lado a lado, dentro de las celdas del campo bajo análisis). Es decir, la máquina deberá recoger todos las lombrices de la "columna" o "fila" que posea la suma de la productividad esperada más alta posible.
Escriba un programa que al darle el mapa de campo de lombrices que describe la productividad estimada en cada una de sus celdas, calcule el número total esperado de lombrices que se recogerá por la máquina durante la prueba (siguiendo el criterio establecido con anterioridad).

Entrada

La primera línea de la entrada contiene dos valores enteros N y M que representan la cantidad de filas (1 ≤ N ≤ 100) y columnas (1 ≤ M ≤ 100), respectivamente, de la granja de células a analizar. A continuación, se ingresan las N filas que contienen M valores enteros que representan las productividades esperadas en cada una de las celdas de la granja. Las entradas deben ser leídas desde el dispositivo de entrada estándar.

Salida

La salida debe estar compuesta de una sola línea que contiene un valor entero que indica el número total previsto de lombrices a ser cosechada por la máquina durante la prueba. Esta salida debe debe ser escrita en el dispositivo de salida estándar.

Ejemplo:

Entrada	**Salida**
3 4 81 28 240 9 40 10 100 240 20 180 110 35	450
4 1 100 110 0 100	310

Problema 9: Matriz Cuadrada

Escriba un algoritmo que lea un número entero N (0 ≤ N ≤ 15) correspondiente al orden de una matriz cuadrada M de enteros y construya una nueva matriz siguiendo el ejemplo que se muestra más abajo.

Entrada

La entrada consiste de valores enteros, uno por línea, correspondientes al tamaño de la matriz a ser construída. La lectura de estos valores finaliza cuando se ingresa el valor 0.

Salida

Para cada valor entero de entrada se imprimirá la matriz correspondiente de acuerdo al siguiente ejemplo. Los valores de las matrices deberán ser formateados en un campo de tamaño T, justificados a la derecha y separados por un espacio en blanco, donde T es igual a la cantidad de dígitos del mayor valor de la matriz. Después del último carácter de cada línea de la matriz no debe haber un espacio en blanco. Además, después de la impresión de cada matriz se debe dejar una línea en blanco.

Ejemplo:

Entrada	Salida
1 2 3 4 5 0	1 1 2 2 4 1 2 4 2 4 8 4 8 16 1 2 4 8 2 4 8 16 4 8 16 32 8 16 32 64 1 2 4 8 16 2 4 8 16 32 4 8 16 32 64 8 16 32 64 128 16 32 64 128 256

Problema 10: Sudoku

El Sudoku se hizo conocido en todo el mundo muy rápidamente, siendo el pasatiempo más popular en el planeta actualmente. Sin embargo algunas personas, completan la matriz incorrecctamente, y no respetan las reglas. Tu tarea es escribir un programa que chequee si una matriz con valores es una solución al Sudoku ó no.
La matriz es una matriz de 9x9 de enteros. Será considerada una solución al Sudoku, si cada fila y cada columna contiene todos los números entre 1 y 9. Además, si la matriz se particiona en 9 submatrices de 3x3 (como se muestra en la figura), cada una de ellas debe también contener todos los números entre 1 y 9. La siguiente matriz es un ejemplo de solución para el Sudoku.

$$\left(\begin{array}{ccc|ccc|ccc} 1 & 3 & 2 & 5 & 7 & 9 & 4 & 6 & 8 \\ 4 & 9 & 8 & 2 & 6 & 1 & 3 & 7 & 5 \\ 7 & 5 & 6 & 3 & 8 & 4 & 2 & 1 & 9 \\ \hline 6 & 4 & 3 & 1 & 5 & 8 & 7 & 9 & 2 \\ 5 & 2 & 1 & 7 & 9 & 3 & 8 & 4 & 6 \\ 9 & 8 & 7 & 4 & 2 & 6 & 5 & 3 & 1 \\ \hline 2 & 1 & 4 & 9 & 3 & 5 & 6 & 8 & 7 \\ 3 & 6 & 5 & 8 & 1 & 7 & 9 & 2 & 4 \\ 8 & 7 & 9 & 6 & 4 & 2 & 1 & 5 & 3 \end{array}\right)$$

Entrada

Se ingresan varias instancias. La primer línea de la entrada contiene n > 0, el número de matrices en la entrada. Las líneas siguientes describen n matrices. Cada matriz está descripta por 9 líneas. Estas líneas contienen 9 enteros cada una.

Salida

Para cada instancia, tu programa debe imprimir una línea conteniendo “Instancia k”, donde k es el número de la instancia. En la segunda línea, tu programa debe imprimir “SIM” (portugués de Si) si la matriz dada es una solución para el Sudoku, ó “NAO” (portugués de No) en otro caso. Imprima siempre una línea en blanco luego de cada instancia.

Entrada	**Salida**
2 1 3 2 5 7 9 4 6 8 4 9 8 2 6 1 3 7 5	Instancia 1 SIM

7 5 6 3 8 4 2 1 9 6 4 3 1 5 8 7 9 2 5 2 1 7 9 3 8 4 6 9 8 7 4 2 6 5 3 1 2 1 4 9 3 5 6 8 7 3 6 5 8 1 7 9 2 4 8 7 9 6 4 2 1 5 3 1 3 2 5 7 9 4 6 8 4 9 8 2 6 1 3 7 5 7 5 6 3 8 4 2 1 9 6 4 3 1 5 8 7 9 2 5 2 1 7 9 3 8 4 6 9 8 7 4 2 6 5 3 1 2 1 4 9 3 5 6 8 7 3 6 5 8 1 7 9 2 4 8 7 9 6 4 2 1 3 5	Instancia 2 NAO