Nivel “AEDD” - Problem Set

18 de Setiembre

Problema 1: Escalator

Código: ESCALATOR

Las escaleras mecánicas han facilitado mucho la vida de las personas. Subir escaleras es una de las tareas más aburridas que se han inventado (después de la invención de las escaleras en primer lugar).

Después de algunas observaciones, te das cuenta de que hay una gran cantidad de energía que se desperdicia en las escaleras mecánicas, porque siguen trabajando incluso cuando no hay nadie usándolas. Para evitar esto, el propietario de un centro comercial local instaló un sensor que verifica cuando hay alguien en la escalera mecánica. Cuando el sensor detecta que no hay nadie, la escalera mecánica se desactiva, de esta forma se ahorra energía hasta que la siguiente persona llega.

Para ser más precisos, el sistema funciona de la siguiente forma: la escalera mecánica está inactiva inicialmente. La cantidad de tiempo que una persona tarda en ir desde el principio hasta el final de la escalera mecánica es de 10 segundos. En otras palabras, si una persona llega a la escalera mecánica en el tiempo t, ésta estará activa en los tiempos t, t + 1, t + 2,..., t + 8 y t + 9, y luego se desactivará en el tiempo t + 10, cuando la persona sale de la escalera mecánica. Esta ventana de tiempo se puede prolongar si una o más personas llegan a la escalera mecánica durante dicho proceso.

El propietario del centro comercial local solicitó tu ayuda. Escribe un algoritmo que, teniendo en cuenta los tiempos en los que algunas personas llegaron a la escalera mecánica, diga cuántos segundos la escalera mecánica estuvo activa.

Entrada

Habrá a lo sumo 30 casos de prueba. Cada caso de prueba comienza con una línea que contiene un entero N, que representa el número de personas que utilizan la escalera mecánica en ese día ($1 \leq N \leq 100$).

En la siguiente línea habrá N números enteros distintos, en orden ascendente, representando el tiempo t en el que cada persona llegó a la escalera mecánica ($1 \leq t \leq 1000$).

El último caso de prueba está indicado con N = 0, el que no debe ser procesado.

Salida

Para cada caso de prueba imprimir una línea que contenga un número entero, representando la cantidad de segundos que la escalera mecánica estuvo activa.

Ejemplo

Entrada	**Salida**
1 5	10
2 12 25	20
2 13 16	13
5 15 20 29 31 50	36
0	

Problema 2: ThreeBonacci

Código: TREEBONACCI

Un número pertenece a la sucesión de TresBonacci si pertenece a la sucesión de Fibonacci (considerar 1 como el primer número de esta sucesión) y satisface al menos uno de los siguientes criterios:

1. La representación numérica de dicho número contiene al menos un dígito 3.
2. El número es un múltiplo de 3.

Entrada

Cada caso de prueba contiene un entero N ($1 \leq N \leq 60$). La entrada termina con EOF.

Salida

Para cada caso de prueba imprimir una sola línea que contenga el N-ésimo número de la sucesión TresBonacci.

Ejemplo

Entrada	Salida
1	3
3	21
EOF	

Problema 3: Random Bingo Cards

Código: RANDOMBINGOCARDS

Usted le pidió a un novato que haga un programa para generar cartones de bingo al azar. El novato dijo que conocía las reglas del bingo y desestimó sus explicaciones.

Adivine ¿cuál fue el resultado de esa prisa? El novato sólo generó 24 números aleatorios distintos (en el intervalo [1, 75] para cada tarjeta), sin respetar el intervalo al que cada número debía pertenecer (B → [1, 15], I → [16, 30], N → [31, 45], G → [46, 60], O → [61, 75]).

Usted decidió crear un programa para juzgar las tarjetas generadas por el programa del novato.

Entrada

Cada caso de prueba contiene una sola línea con 24 números enteros separados por espacios. Para la tarjeta de la imagen, la entrada sería la secuencia: 15 28 36 49 65 13... 53 69. La entrada termina con EOF.

Salida

Para cada tarjeta analizada imprimir una sola línea que contenga uno de los tres posibles veredictos: "OK", si la tarjeta es válida; "RECICLAVEL" (reciclable), si es posible encontrar alguna permutación que haga que la tarjeta sea válida; o "DESCARTAVEL" (desechable), en caso que tal permutación no exista.

Ejemplos

Entrada	**Salida**
15 28 36 49 65 13 22 45 59 72 1 20 47 71 6 19 43 56 75 5 29 31 53 69	OK
69 28 36 49 65 13 22 45 59 72 1 20 47 71 6 19 43 56 75 5 29 31 53 15	RECICLAVEL
15 28 36 49 65 13 22 45 59 72 1 20 41 71 6 19 43 56 75 5 29 31 53 69	DESCARTAVEL

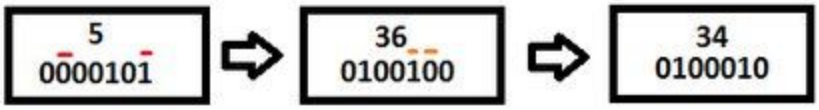
Problema 4: Bit Shuffling

Código: BITSHUFFLING

Para saber si sus estudiantes realmente entendieron la clase sobre la representación binaria de los números enteros, el profesor llegó con el siguiente problema:

"Teniendo en cuenta un número entero y una secuencia de permutaciones de bits por su representación binaria, encuentre 3 números: el número resultante después de todas las permutaciones, el valor máximo y el valor mínimo encontrados durante las permutaciones".

El profesor prometió un punto extra a quien resolviera el problema primero. Dado que él nunca hizo tal cosa antes (dar puntos extra), los estudiantes se apresuraron a resolver el problema lo más rápido posible, temiendo que el profesor pudiera cambiar de opinión.



Entrada

La primera línea de un caso de prueba contiene los enteros **N** ($0 \leq N \leq 2^{32} - 1$) y **K** ($0 \leq K \leq 100$), que representan el número inicial y el número de permutaciones, respectivamente. Cada una de las siguientes **K** líneas contiene dos enteros separados por espacios **A** y **B** ($0 \leq A, B \leq 31$), indicando que los bits **A** y **B** deben ser intercambiados. La entrada termina cuando **N** = **K** = 0.

Salida

Para cada caso de prueba imprimir 3 enteros separados por un espacio: **RES MAX MIN**, donde **RES** es el número **N** alcanzado después de haber realizado todas las permutaciones, **MAX** y **MIN** son, respectivamente, el valor intermedio más grande y el más pequeño encontrado en el proceso de permutación (**MAX** y **MIN** deben considerar también el número inicial y **RES**).

Ejemplo

Entrada	Salida
5 2 0 5 1 2 0 0	34 36 5

Problema 5: Sum of Two Squares

Código: SUMOFTWOSQUARES

¿Qué números enteros se pueden representar por una suma de dos enteros al cuadrado?

¡Esa es la pregunta que su programa debe responder!

Por ejemplo, el número 41 se puede representar como $(-4)^2 + 5^2 = 41$, pero 7 no se puede representar de la misma manera.

Entrada

La entrada consiste de varias líneas, cada línea contiene un número entero con valor absoluto inferior o igual a 10.000. La entrada de datos termina con EOF.

Salida

Para cada línea imprimir "YES" (SÍ) si el número se puede representar por una suma de dos números enteros al cuadrado, de lo contrario imprimir "NO".

Ejemplos:

Entrada	**Salida**
41	YES
7	NO
2	YES

Problema 6: Focus

Código: VARYINGFOCUS

Daniel está tomando un curso de Visión Computacional y decidió reproducir una obra interesante que vio en clase: tomó algunas fotos de una misma escena, variando sólo el foco, para después combinarlas en una sola foto, en la que todos los objetos en escena están en foco simultáneamente. Para eso, necesita que cada objeto se vea nítidamente en al menos una foto.

Daniel sabe que, para cada objeto, hay un intervalo cerrado de los planos focales en los que se mantiene dicho objeto claramente visible. Por ejemplo, en la imagen de abajo, (i), (ii) y (iii) son tres fotos de la misma escena, cada una tomada con un foco diferente; (iv) es la imagen combinada generada por Daniel.

Como la tarjeta de memoria de la cámara de Daniel tiene una pequeña capacidad, pidió su ayuda. Teniendo en cuenta los intervalos de foco de todos los objetos en la escena que Daniel pretende fotografiar, determine el número mínimo de fotografías que él debe tomar para que cada objeto quede nítido en al menos una de las fotos.

Entrada

La entrada consiste en varios casos de prueba. La primera línea de cada caso de prueba contiene un entero **N** ($1 \leq N \leq 10^6$) que indica el número de objetos en la escena. Cada una de las siguientes **N** líneas contiene dos números enteros, **A** y **B** ($1 \leq A \leq B \leq 10^9$), que indican los extremos del intervalo de foco de cada objeto. La entrada termina con EOF.

Salida

Para cada caso de prueba debe imprimir una línea con un entero que indique el número mínimo de fotografías que Daniel tiene que tomar.

Ejemplos:

Entrada	**Salida**
3 1 3 2 5 4 6	2
5 1 2	3

5 6 3 4 5 6 1 2	

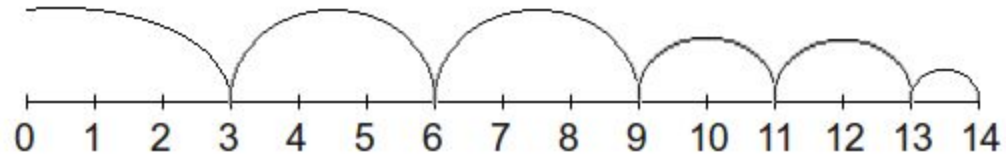
Problema 7: Throwing Balls

Código: THROWINGBALLS

Sus amigos inventaron un nuevo concurso: Lanzar pelotas. El objetivo es sencillo, basta con lanzar una pelota de manera que caiga en un agujero **N** metros más adelante.

Cuando se lanza la pelota, digamos que a una velocidad de **V** enteros, se queda en el aire por **V** metros y luego toca el suelo. Este proceso se repite **V** veces. Después que la pelota toca el suelo **V** veces, cambia su velocidad a **V** – 1, y el proceso anterior se repite, hasta que la velocidad es igual a 0.

Por ejemplo, si la pelota es lanzada a una velocidad igual a 3, tocará el suelo en los puntos siguientes: 3, 6, 9, 11, 13 y 14, como se puede ver en la siguiente imagen.



Se puede lanzar una pelota a una velocidad que es un entero menor o igual a **V**. Dada la distancia del agujero, diga si es posible para usted tirar la pelota y que ésta caiga exactamente en el agujero.

Entrada

Habrá varios casos de prueba. Cada caso de prueba contiene dos números enteros, **N** y **V** ($1 \leq N \leq 1000$, $1 \leq V \leq 30$), que representan la distancia del agujero y la velocidad máxima a la que se puede lanzar la pelota, respectivamente.

El último caso de prueba está indicado cuando N = V = 0, que no debe ser procesado.

Salida

Para cada caso de prueba, imprimir una línea que contenga la palabra "possivel" (sin las comillas) si es posible lanzar la pelota a una velocidad que es un entero menor o igual a **V**, de manera que la pelota caiga en el agujero, o "impossivel" en caso contrario.

Ejemplos:

Entrada	**Salida**
14 3	possivel
13 3	possivel
12 3	impossivel
5 3	possivel
30 4	possivel
0 0	

Problema 8: Contamination

Código: CONTAMINATION

El año es 2241 y la colonización de otros planetas es ya una realidad. Usted trabaja en el centro de control de los recursos del planeta URI-942, principalmente controlando el suministro de agua. El agua se almacena en tanques subterráneos, protegidos de las altas temperaturas de la superficie.

Pero sus colegas, Ana y Mario, encontraron algunas deficiencias en las paredes de los tanques, que pueden conducir a la contaminación del agua almacenada. Sus colegas fueron capaces de identificar los puntos donde puede haber defectos con la infiltración de contaminantes. Sabiendo que los contaminantes se extienden a lo largo del tanque de agua afectado, su tarea consiste en estimar la contaminación del agua de acuerdo con los mapas proporcionados por sus pares.

Los mapas se discretizaron en celdas, y las celdas pueden corresponder a una región de roca, agua (tanque) o un contaminante. Debido a las grietas, una celda con contaminante contamina las celdas adyacentes (izquierda, derecha, arriba y abajo) que contienen agua, pero la contaminación está bloqueada por las celdas de roca.

Entrada

La entrada consiste de varios mapas, y una descripción de cada mapa se inicia con una línea que contiene dos enteros **N** y **M**, que corresponden al número de filas y columnas del mapa. Las siguientes **N** líneas describen el mapa; cada línea contiene **M** caracteres hasta el salto de línea. Los caracteres posibles son: **A**, que representa una celda que contiene agua; **X**, que representa una celda con roca, y **T**, que representa una celda con contaminante.

La entrada termina cuando **N** = **M** = 0; este caso no debe ser procesado. En todos los mapas, **N** y **M** son menores que o iguales a 50.

Salida

Para cada mapa, imprimir una estimación de la contaminación futura. Esta estimación debe coincidir con el mapa original (como se ve en la entrada), pero sustituyendo las celdas con agua que han sido contaminadas por el carácter **T**. Deje una línea en blanco después de cada mapa (incluyendo el último mapa).

Ejemplo:

Entrada	**Salida**
6 7 XXAAXXX XXAAXAX XXXXAXX XAAAAAX TAAXAAA XXXXXXX 3 3	XXAAXXX XXAAXAX XXXXTXX XTTTTTX TTTXTTT XXXXXXX TTT

TTT XXX AAA 0 0	XXX AAA