**Mensajería**

Tiempo límite por caso: 1 seg

Memoria límite por caso: 32 MB

Andrea, la matemágica alfa, administra toda la mensajería ultra secreta de la logia de los matemágicos. Como Andrea es muy buena en su trabajo, quiere minimizar el costo de enviar un mensaje a todos los cuarteles de la logia.

Andrea sabe que la logia tiene **N** cuarteles ultra secretos en el estado de Querétaro y que están numerados de **0** a **N−1**. También sabe que las rutas para enviar los mensajes van de cuartel a cuartel y funcionan de la siguiente manera:

* El cuartel 0 tiene una ruta de envío con el cuartel 1.
* El cuartel 1 tiene una ruta de envío con el cuartel 2.
* Y así sucesivamente hasta el cuartel **N−1** que tiene una ruta de mensajería con el cuartel 0.

Las rutas de envío de un cuartel a otro tienen un costo (**ai**) y funcionan en ambos sentidos, es decir, la ruta del cuartel 0 al cuartel 1 también funciona del cuartel 1 al cuartel 0 con el mismo costo.

Para minimizar los costos, Andrea puede poner centrales de envío en cualquier cuartel y enviar el mensaje desde todas las centrales disponibles al mismo tiempo.

Poner una central de envío tiene un costo **K**.

Sin embargo, la ventaja de poner nuevas centrales de envío en otros cuarteles es que estos cuarteles ahora pueden enviar mensajes a los demás cuarteles sin ningún costo adicional más que el de las rutas usadas.

Para poder enviar el mensaje, Andrea siempre debe poner al menos la primera central de envío en alguno de los cuarteles.

# Problema

Ayuda a Andrea a minimizar el costo total de enviar un mensaje a todos los cuarteles de la logia.

# Entrada

En la primera línea se encuentran los enteros **N** y **K**. En las siguientes **N** líneas hay un entero **ai**representando el costo de la ruta de mensajería del cuartel **i** al cuartel **i+1**.

# Salida

La suma que minimice el costo de enviar un mensaje, considerando el costo de poner las nuevas centrales de envío y el costo de todas las rutas usadas para enviar el mensaje a toda la logia.

# Ejemplo

| **Entrada** | **Salida** | **Descripción** |
| --- | --- | --- |
| 4 5  2  6  3  8 | 15 | Andrea pone una central en el cuartel 1 y en el cuartel 3, cada una con costo **K** (5 para este caso). Luego para enviar el mensaje al cuartel 0, se toma la ruta desde el cuartel 1 (con costo 2, recuerda que las rutas son bidireccionales y el costo de enviar de 0 a 1 es el mismo que de 1 a 0) y para el cuartel 2 se toma la ruta desde el cuartel 3 (con costo 3), los cuarteles 1 y 3 reciben el mensaje sin costo por tener una central de envío. La suma total del costo es 15 = 5 + 5 + 2 + 3 |
| 2 3  2  5 | 5 | Andrea pone una central en el cuartel 0 y de éste se toma la ruta al cuartel 1. El costo de poner una central en el cuartel 0 es de 3. De ahí se envía al cuartel 1 utilizando la ruta que va de 0 a 1 la cual tiene un costo 2. La suma del costo total es 3 + 2 = 5 |

# Límites

* 2 ≤ **N** ≤ 1,000,000
* 1 ≤ **ai,K** ≤1,000
* Para un 36% de los casos, **N**≤1,000

**Truco Asombroso**

Tiempo límite por caso: 1 seg

Memoria límite por caso: 32 MB

Ethan el matemágico, ha empezado a armar su nuevo truco de magia y matemáticas. Este nuevo truco hará que Ethan se convierta en el más famoso entre todos los matemágicos del mundo. Tú, como mejor amigo de Ethan, has decidido ayudarle para que su acto no tenga fallas y logre su fama.

El truco de Ethan consiste en agarrar un número y modificarlo varias veces con alguno de los siguientes actos:

* Sumarle 1 al número que tiene.
* Restarle 1 al número que tiene.
* Dividir entre 2 siempre y cuando el número que tiene sea par.

El truco termina cuando el número que tiene sea 1.

# Problema

Tu tarea es ayudar a Ethan a ejecutar su truco en el mínimo número de actos para que el truco sea asombroso.

Para hacer el truco más asombroso, el número inicial está expresado en notación binaria.

# Entrada

En la primera línea el entero **N** que representa la cantidad de bits (cifras) del número binario inicial. En la segunda línea una cadena de longitud **N** que contiene 0s y 1s (sin separación). Esta cadena representa el número inicial. El primer carácter es el bit más significativo.

# Salida

Un entero que indique la mínima cantidad de actos que tiene que hacer Ethan para que el número original se convierta en 1.

# Ejemplo

| **Entrada** | **Salida** | **Descripción** |
| --- | --- | --- |
| 2  11 | 2 | 3->2->1  El número inicial (en notación decimal) es 3.  La primera operación es restarle 1 con lo que queda como 2.  La segunda operación es dividirlo entre 2 con lo que se obtiene un 1 y el acto ha terminado después de 2 pasos. |
| 4  1010 | 4 | 10->5->4->2->1  El número inicial en notación decimal es 10.  Se divide entre 2 para obtener 5, se le resta 1 para obtener 4, se divide entre 2 para obtener 2, se divide entre 2 para obtener 1. El acto ha terminado después de 4 pasos. |

# Consideraciones

* 2 ≤ **N** ≤ 106
* Para un 27% de los casos 2 ≤ **N** ≤20

**Convención**

Tiempo límite por caso: 1 seg

Memoria límite por caso: 32 MB

Alexis y varios otros de sus amigos matemágicos se encuentran en la Convención Nacional de Matemáticas y Magia. Alexis, como quiere fomentar que sus amigos compartan su magia y conocimiento, se le han ocurrido algunas ideas para que todos convivan más.

Una de las ideas de Alexis es juntar al grupo más grande de amigos que se conozcan todos entre sí. Esto es, que cada persona del grupo sea amigo de todas y cada una de las otras personas en ese mismo grupo.

Esta es una tarea muy fácil para Alexis y las tareas fáciles no son cool, por lo que Alexis quiere añadir una restricción a esta idea. Para ello, lo que va a hacer Alexis es numerar del 0 al **N−1** a las **N** personas de la convención. Después va a juntar al grupo más grande de amigos que se conozcan todos entre sí, de tal forma que en ese grupo de personas sus números sean consecutivos.

# Problema

Ayuda a Alexis a compartir la magia y el conocimiento juntando al grupo más grande de personas que cumplan con la restricción de su idea cool (recuerda que la restricción cool es que todos los miembros de un grupo, además de ser amigos entre sí, tengan números consecutivos).

# Entrada

En la primera línea los enteros **N** y **M**, representando el número de asistentes a la convención y el número de amistades respectivamente. Cada una de las siguientes **M** líneas tendrá dos enteros **ai bi** que indican que la persona **ai** es amiga de la persona **bi** y viceversa.

# Salida

Un entero representando el tamaño del grupo más grande de personas que cumplan con la restricción cool de Alexis.

# Ejemplo

| **Entrada** | **Salida** | **Descripción** |
| --- | --- | --- |
| 4 4  0 3  1 2  0 2  1 0 | 3 | Las personas 0, 1 y 2 forman el grupo más grande de amigos que se conocen todos entre sí y que además son números consecutivos. |
| 2 1  0 1 | 2 |  |

# Consideraciones

* 2 ≤ **N** ≤1,000,000
* 1 ≤ **M** ≤200,000
* Para un 20% de los casos, 1 ≤ **N** ≤ 200
* Para un 40% de los casos, 1 ≤ **N** ≤1000
* Para un 40% de los casos, 1 ≤ **M** ≤1000
* No hay personas que sean amigas de sí mismas
* No hay amistades repetidas

**Atrapando a Karel**

Tiempo límite por caso: 1 seg

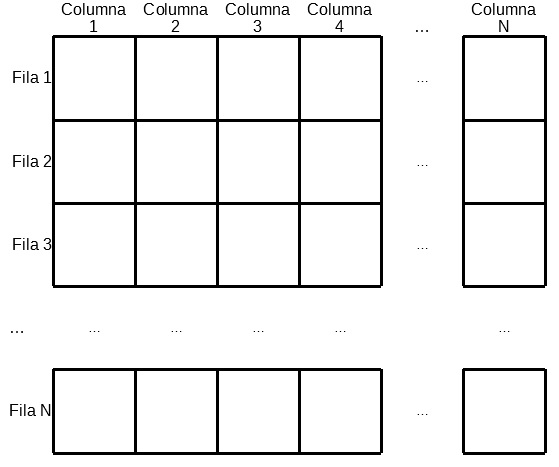
Memoria límite por caso: 32 MB

Se han escuchado rumores de que Karel está muerto, pero en la logia de los matemágicos están convencidos de que eso no es verdad.

La realidad es que Roque el filósofo, uno de los maléficos archienemigos de los matemágicos, lo tiene encerrado en una cuadrícula y solo lo deja salir durante la OMI de primaria y secundaria.

Ante esto Karel no se ha rendido y hará todo lo posible por escapar. Roque el filósofo descubrió su plan y decidió abusar de su autoridad intimidándote y exigiéndote que evites que escape y lo castigues encerrándolo en una casilla.

La cuadrícula en la que Karel está encerrado tiene **N** filas numeradas de 1 a **N** (de arriba hacia abajo), y **N** columnas, también numeradas de 1 a **N** (de izquierda a derecha).



Inicialmente Karel se encuentra en el centro de la cuadrícula, **N** siempre será impar por lo que siempre habrá una única casilla central, además no hay muros inicialmente en la cuadrícula.

En cada segundo Karel se puede mover a cualquiera de las 4 casillas adyacentes (o salir de la cuadrícula si se encuentra en una de las casillas que están en el borde de la misma) siempre y cuando no haya un muro obstruyendo su camino (esta vez Karel tiene prisa y no necesita gastar tiempo girando para empezar a caminar, por lo tanto puedes considerar que Karel puede girar sin consumir tiempo).

En cada segundo puedes agregar un muro en cualquier lugar de la cuadrícula. Tienes la ventaja de que el primer muro que pongas precede al primer movimiento de Karel (es decir, tú pones un muro, luego Karel se mueve una casilla, pones un segundo muro, Karel se mueve a otra casilla, etc.).

# Problema

Escribe un programa que atrape a Karel en una casilla (un rectángulo de 1x1) y evites así ser castigado por Roque el filósofo.

Recuerda que estamos en tiempos de austeridad, así que entre menos muros utilices para atrapar a Karel más puntos obtendrás.

# Entrada y Salida

**Este es un problema interactivo**, por lo que no tendrás que leer la entrada ni imprimir la salida, sino implementar en tu código la función atrapando y mandar llamar la función del evaluador ponMuro para completar tu tarea.

**Para obtener más información sobre los detalles de implementación de este problema debes revisar el texto del problema en la plataforma OmegaUp.**

# Implementación

## Tu función atrapando()

C/C++ void atrapando(int N);

Pascal procedure atrapando(var N: LongInt);

### **Descripción**

El evaluador buscará en tu código esta función y la llamará con el número N como parámetro. Tu implementación deberá llamar a la función ponMuro para poner un muro en una casilla en específico, y esta misma función te dirá hacia dónde se mueve Karel.

### **Parámetros**

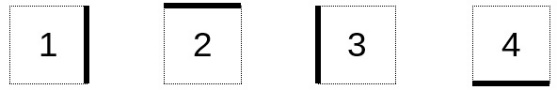
* **N**: Un entero indicando el tamaño de la cuadrícula.

## Función del evaluador ponMuro()

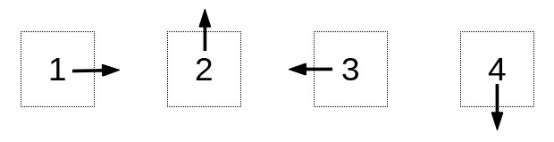
C/C++ int ponMuro(int f, int c, int d);

Pascal function ponMuro(var f, c, d: LongInt):LongInt;

Esta función coloca un muro al lado de la casilla que se encuentra en fila f y columna c. El parámetro d indica con un número del 1 al 4, en cuál de las 4 aristas (de las que se encuentran alrededor de esa casilla) poner el muro:



Luego de poner el muro, Karel se moverá en alguna de las 4 direcciones. La función regresará un entero del 1 al 4 indicando hacia cuál de las 4 direcciones se movió:



La función regresará 0 en caso de que Karel se encuentre atrapado en la celda.

# Rutina de Ejemplo

A continuación, se muestran las primeras llamadas de una rutina de ejemplo.

| **Entrada** | **Salida** | **Descripción** |
| --- | --- | --- |
| Función llamada | Valor devuelto | Descripción |
| atrapando(21) | - | Esta será la llamada inicial a tu procedimiento atrapando. El mundo mide 21x21 y por lo tanto Karel se encuentra en la casilla (11,11) |
| ponMuro(11,11,2) | 1 | Muro fue colocado al norte de la casilla (11,11). Karel se mueve al este a la casilla (11,12). |
| ponMuro(11,12,1) | 4 | Muro fue colocado al este de la casilla (11,12). Karel se mueve al sur a la casilla (12,12). |
| ponMuro(12,12,4) | 1 | Muro fue colocado al sur de la casilla (12,12). Karel se mueve al este a la casilla (12,13). |
| ponMuro(12,13,1) | 4 | Muro fue colocado al este de la casilla (12,13). Karel se mueve al sur a la casilla (13,13). |

# Restricciones

20 < **N** < 500

# Consideraciones

* Debes encerrar a Karel en una casilla para atraparlo
* Siempre hay al menos una forma de atrapar a Karel
* Si Karel sale de la cuadricula obtendrás 0 puntos en ese caso
* Si mandas llamar más de 100**N** veces la función ponMuro obtendrás 0 puntos
* Se garantiza que siempre recibirás una **N** para la cual será posible atrapar a Karel
* Mientras menos muros pongas obtendrás más puntos
* **N** siempre será un número impar