Carrera Campo Traviesa

|  |  |  |  |
| --- | --- | --- | --- |
| Puntos |  | Límite de memoria | 64MB |
| Límite de tiempo (caso) | 1s | Límite de tiempo (total) | 60s |

**Historia**

Karel se inscribió en una carrera de campo traviesa. La carrera se corre en un circuito, cada sección del circuito tiene un nivel de dificultad. Los participantes deben iniciar en algún punto del circuito y recorrer las siguientes ***K*** secciones del mismo. **Un corredor siempre recorre *K* secciones a partir de donde empiece, no importa el largo total del circuito.**

La *dificultad* total del recorrido se calcula sumando el nivel de dificultad de las ***K*** secciones. Para un circuito, la dificultad total varía dependiendo de la posición dónde se inicia la carrera. Karel es muy competitivo y le gustan los retos, por lo tanto quiere buscar la **dificultad total máxima** posible del circuito.

En el mundo de Karel está dibujado el circuito con montones de zumbadores. Cada montón es una sección. El número de zumbadores en el montón representa la dificultad de esa sección. El mundo es rectangular sin paredes intermedias. El circuito puede tener cualquier forma pero puedes estar seguro de que ninguna sección toca la pared y que cada sección del circuito está conectada exactamente a otras dos secciones. El número ***K*** de secciones a recorrer en el circuito estará representado por un montón de zumbadores en la posición (1,1) del mundo.

**Problema**

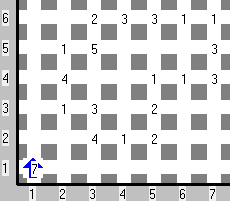
Escribe un programa que dado un circuito y el número ***K*** de secciones a recorrer le ayude a Karel a determinar la **dificultad total máxima posible** del mismo. Tu programa deberá dejar un montón de zumbadores en la posición (1,1) igual a la dificultad total máxima del circuito.

**Consideraciones**

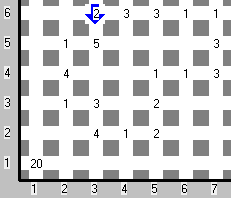
* Karel inicia en la posición (1,1) orientado al norte.
* El número ***K*** de secciones a recorrer está representado por un montón en la posición (1,1) y será un valor entre 1 y 100.
* Karel inicia con **infinitos zumbadores en la mochila**.
* El mundo de Karel siempre será un cuadro de 100 filas por 100 columnas.
* El circuito nunca tendrá más de 160 secciones de largo.
* No importan la dirección ni la orientación final de Karel, únicamente los zumbadores que dejes en la posición (1,1)

**Ejemplo**

**input**



**output**



El mundo de ejemplo muestra un circuito y el número **K=7** de secciones a recorrer en la posición (1,1). En el mundo de salida Karel dejó un montón de 20 zumbadores en la posición (1,1) ya que esa es la **dificultad total máxima** del circuito. En el mundo se muestra también dónde tiene que iniciar el recorrido para lograr esa dificultad. **La posición de Karel NO IMPORTA, sólo el montón de zumbadores que representa la dificultad total máxima del circuito.**

Karel-ccionista

|  |  |  |  |
| --- | --- | --- | --- |
| Puntos |  | Límite de memoria | 32MB |
| Límite de tiempo (caso) | 1s | Límite de tiempo (total) | 60s |

**Descripción**

Karel es un ávido coleccionista de tarjetas de súper héroes. Ayer decidió desempolvar su colección de los **Karel-Vengers**. Al abrir la caja dónde las tenía descubrió que las tarjetas están desordenadas e incompletas **:(**.

La colección de tarjetas completa tiene 100 tarjetas numeradas del 1 al 100.

Resignado a que algunas tarjetas se le han perdido, Karel quiere saber 2 cosas:

* Si tiene tarjetas repetidas.
* Si las tarjetas que todavía le quedan son continuas en numeración.

En la primera fila del mundo de Karel, iniciando en la columna 1 habrá montones de zumbadores. Cada montón representa el número de una tarjeta. Los números pueden ir desde 1 hasta 100. Las tarjetas están una junto de otra, es decir, no hay espacios con 0 zumbadores entre las tarjetas.

**Problema**

Escribe un programa que, dada la lista de tarjetas verifique si Karel tiene tarjetas repetidas y si las tarjetas de Karel están continuas en numeración.

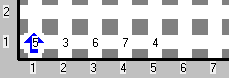
Si Karel tiene **REPETIDAS** o sus tarjetas **NO SON CONTINUAS** entonces deberá apagarse orientado al **SUR**.

Si Karel **NO TIENE REPETIDAS** y sus tarjetas **SON CONTINUAS EN NUMERACIÓN** entonces deberá apagarse orientado al **NORTE**.

**Consideraciones**

* Karel empieza en la posición (1,1) orientado al norte.
* Karel inicia con 0 zumbadores en su mochila.
* La primera tarjeta siempre está en la posición (1,1).
* El mundo mide 100 filas por 100 columnas y no tiene paredes internas.
* Para obtener puntos **Karel debe apagarse viendo al SUR si hay REPETIDAS o NO SON CONTINUAS u orientado al NORTE si NO HAY REPETIDAS y todas SON CONTINUAS**.
* No importan la posición final de Karel ni los zumbadores que dejes en el mundo.
* En este problema los casos se agruparán de modo que cada grupo contenga al menos un caso cuya solución es terminar orientado al norte y un caso cuya solución sea terminar orientado al sur.

**Ejemplo1**



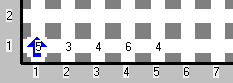
En este ejemplo Karel tiene las tarjetas **3, 4, 5, 6 y 7**. **NO HAY REPETIDAS** y **TODAS SON CONTINUAS**. Por lo tanto Karel debe apagarse orientado al **NORTE**.

**Ejemplo2**



En este ejemplo Karel tiene las tarjetas **3, 4, 5, 7 y 8**. **NO HAY REPETIDAS** pero **FALTA LA TARJETA 6**. Por lo tanto Karel debe apagarse orientado al **SUR**.

**Ejemplo3**



En este ejemplo Karel tiene las tarjetas **3, 4, 4, 5 y 6**. La tarjeta **4** está **REPETIDA**. Por lo tanto Karel debe apagarse orientado al **SUR**.

Karel Anti-cuadros

|  |  |  |  |
| --- | --- | --- | --- |
| Puntos |  | Límite de memoria | 64MB |
| Límite de tiempo (caso) | 1s | Límite de tiempo (total) | 30s |

**Historia**

Karel está experimentando con su lado artístico. Está OMI está decidido a hacer un gran mural, pero Karel tiene una fobia hacia los cuadros y desea que su pintura no contenga ningún cuadro.

El mundo de Karel representa un lienzo en blanco. El mundo es un rectángulo sin paredes intermedias y sin zumbadores.

Si en cualquier lugar del lienzo es posible encontrar un conjunto de posiciones sin zumbadores que formen un cuadrado, se considera que la pintura tiene un cuadro. Al inicio ninguna posición tiene zumbadores, por lo tanto, al inicio la pintura contiene muchos cuadros.

Karel piensa poner zumbadores en algunas posiciones del lienzo para que este no contenga ningún cuadro. Karel quiere poner los menos zumbadores posibles para eliminar todos los cuadros de la pintura. Ayuda a Karel a lograrlo.

**Es importante que revises el ejemplo para que te quede claro cómo son los cuadros que desea eliminar Karel.**

**Problema**

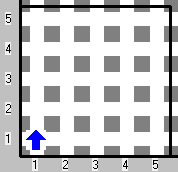
Escribe un programa que dado el lienzo inicial coloque zumbadores dentro del mismo de modo que la pintura no contenga cuadros.

Hay muchas formas de lograr el objetivo anterior, la mejor solución es aquella que coloca **el menor número de zumbadores**, sin embargo, para obtener puntos, no es necesario que tu solución sea la menor posible.

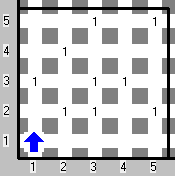
**Consideraciones**

* Karel inicia en la posición (1,1) del mundo orientado al norte.
* Karel inicia con infinitos zumbadores en la mochila.
* El mundo es un cuadrado sin paredes internas. No importa la posición ni la orientación final de Karel.
* Para obtener puntos, tu programa deberá colocar zumbadores de modo que **sea imposible formar cuadros**, buscando siempre colocar la menor cantidad de zumbadores posible.
* **IMPORTANTE**
  + **El número de puntos que obtengas en este problema dependerá de la cantidad de zumbadores que coloques**.
  + Si al terminar la ejecución, la pintura contiene algún cuadro, obtendrás **0 puntos en ese caso**.
  + Si tu programa coloca zumbadores en más del 50% de las posiciones obtendrás **0 puntos**.
  + Para un conjunto de casos con valor de 21 puntos tu programa deberá colocar zumbadores en a lo más el 50% de las posiciones.
  + Para otro conjunto de casos con valor de 23 puntos tu programa deberá colocar zumbadores en a lo más el 35% de las posiciones.
  + Para otro conjunto de casos con valor de 27 puntos tu programa deberá colocar zumbadores en a lo más el 31% de las posiciones.
  + Para un último conjunto de casos con valor de 29 puntos tu programa obtendrá una cantidad de puntos determinada por la siguiente regla:
    - 0% si tu programa coloca zumbadores en más del 31% de las posiciones
    - 100% si tu programa coloca el número mínimo de zumbadores posible.
    - Si tu programa coloca una cantidad de zumbadores que está entre el mínimo posible y el 30% obtendrá una cantidad de puntos proporcional.

**Ejemplos**

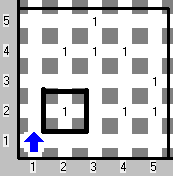


El mundo de entrada es un lienzo de 5 x 5 que inicialmente no tiene zumbadores. En total hay 25 posiciones.

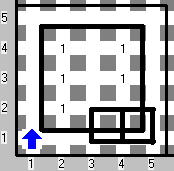


Este es un ejemplo posible de salida, todos los cuadros, de todas las longitudes han sido eliminados. En total se colocaron 9 zumbadores lo que corresponde a un 36% de las posiciones originales. Para este lienzo es posible encontrar arreglos que quitan todos los cuadros colocando únicamente 6 zumbadores lo que representa el 24% de las posiciones originales.

A continuación se muestran algunos ejemplos de resultados inválidos que no eliminan todos los cuadros del lienzo.



En la imagen se muestra que aún hay un cuadro de 2 x 2 que puede formarse.



Esta solución deja dos cuadros de 1 x 1 y uno de 4 x 4 y por tanto es inválida.

**Nota**

En la parte de hasta arriba hay un link para validar tus soluciones con la versión en línea de Karel (karel.js). Te recomendamos utilizarlo para que no pierdas tiempo validando si tus soluciones tienen o no cuadros.

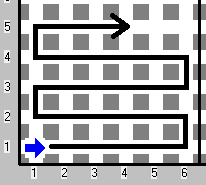
**Minas**

**Historia**

Karel fue seleccionado para el servicio militar en el batallón de ingenieros zapadores. Parte de su entrenamiento es aprender a recorrer campos minados (y salir vivo, por supuesto).

El día de hoy, Karel debe recorrer un campo minado. El campo se representa como un mundo rectangular que no tiene paredes internas. Las minas se representan como montones de zumbadores, la cantidad de zumbadores en el montón determina **el segundo en el que explotará la mina**, es decir, un montón de 5 zumbadores representa una mina que explotará en el segundo 5 a partir de que se inicia el ejercicio. Si al momento en que explota la mina Karel se encuentra **sobre ella, en la casilla de la izquierda o en la casilla de la derecha**, Karel explotará. Cuando una mina explota **no detona** ninguna de las minas a su alrededor.

Karel inicia el recorrido en la posición (1,1) en el segundo cero. Siempre debe recorrer el campo fila por fila como se muestra en la figura:

[](https://github.com/ComiteMexicanoDeInformatica/OMI-2015/blob/master/dia1/minas/statements/bombas1.png)

Karel avanza 1 casilla por segundo, no importa la dirección en la que se esté moviendo. Por ejemplo, inicia en la casilla (1,1) en el segundo cero, en el segundo uno estará en la columna 2 de la fila 1 y para el ejemplo de la figura en el segundo 6 estará en la columna 6 de la fila 2.

**Problema**

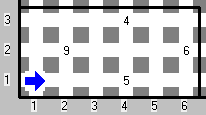
Escribe un programa que reciba el mundo que representa el campo minado y que ayude a Karel a saber si sobrevivirá o no al recorrido. Si Karel explota durante el recorrido, tu programa deberá apagarlo en la posición donde estaba cuando murió. Si Karel sobrevive al recorrido deberá apagarse en la última posición del recorrido. Para que te quede más claro te recomendamos revisar el ejemplo.

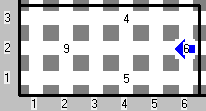
**Consideraciones**

* Karel inicia en la posición (1,1) del mundo orientado al Este.
* Karel inicia con **0 zumbadores en la mochila**.
* No importan la orientación final de Karel, ni los zumbadores que queden en el mundo. **Sólo importa la posición final de Karel**.
* En este problema se agruparán casos de prueba, para obtener los puntos de un grupo, tu programa deberá resolver de manera correcta **todos los casos del grupo**.
* **IMPORTANTE**
  + Para un conjunto de casos con valor de 33 puntos tu programa deberá ejecutar la instrucción **avanza/move** un máximo de **N+1** veces (donde **N** es la cantidad de segundos que se requiere para hacer el recorrido completo). Es decir, si en alguno de estos casos tu programa ejecuta la instrucción **avanza/move** más de **N** veces **obtendrás 0 puntos**.
  + Para el resto de los casos de prueba **no tienes límite en la cantidad de veces que puedes ejecutar la instrucción avanza/move**.

**Ejemplos**

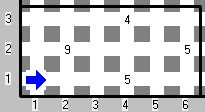
**Ejemplo 1**

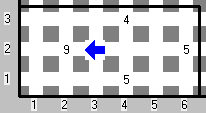
Entrada [](https://github.com/ComiteMexicanoDeInformatica/OMI-2015/blob/master/dia1/minas/statements/entrada1.png)

Salida [](https://github.com/ComiteMexicanoDeInformatica/OMI-2015/blob/master/dia1/minas/statements/salida1.png)

La primera mina por la que pasa Karel es la que está en la fila 1, columna 4. Esa mina explota en el segundo 5. Cuando esa mina explota Karel no está sobre ella ni a su izquierda ni a su derecha. La segunda mina es la de la fila 2, columna 6. Esta mina explota en el segundo 6, justo cuando Karel pasa sobre ella, por lo tanto Karel explota y debe apagarse en esa posición.

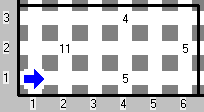
**Ejemplo 2**

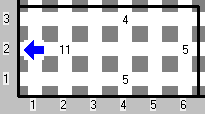
Entrada [](https://github.com/ComiteMexicanoDeInformatica/OMI-2015/blob/master/dia1/minas/statements/entrada2.png)

Salida [](https://github.com/ComiteMexicanoDeInformatica/OMI-2015/blob/master/dia1/minas/statements/salida2.png)

El mundo es similar al anterior, pero esta vez la segunda mina explota en el segundo 5, en ese momento Karel no está sobre ella ni a su izquierda ni a su derecha. La tercera mina (fila 2, columna 2) explota en el segundo 9. En ese momento Karel está a su derecha y por lo tanto explota. **Karel debe apagarse en la posición donde se encontraba al momento de la explosión.**

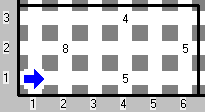
**Ejemplo 3**

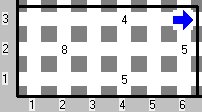
Entrada [](https://github.com/ComiteMexicanoDeInformatica/OMI-2015/blob/master/dia1/minas/statements/entrada3.png)

Salida [](https://github.com/ComiteMexicanoDeInformatica/OMI-2015/blob/master/dia1/minas/statements/salida3.png)

Esta vez la tercera mina explota cuando Karel está a su izquierda. Por lo tanto Karel debe a la izquierda de la mina.

**Ejemplo 4**

Entrada [](https://github.com/ComiteMexicanoDeInformatica/OMI-2015/blob/master/dia1/minas/statements/entrada4.png)

Salida [](https://github.com/ComiteMexicanoDeInformatica/OMI-2015/blob/master/dia1/minas/statements/salida4.png)

En este último ejemplo Karel sobrevive al recorrido sin que ninguna mina explote sobre o junto a él y por lo tanto debe apagarse en la última posición del recorrido.