Robot Contest

Los investigadores de Inteligencia Artificial de la Universidad de Szeged están llevando a cabo una competencia de programación de robots. Tu amigo, Hanga, ha decidido participar en la competencia. El objetivo es programar el mejor *Pulibot*, admirando la gran inteligencia de la famosa raza de perro pastor húngaro, el Puli.

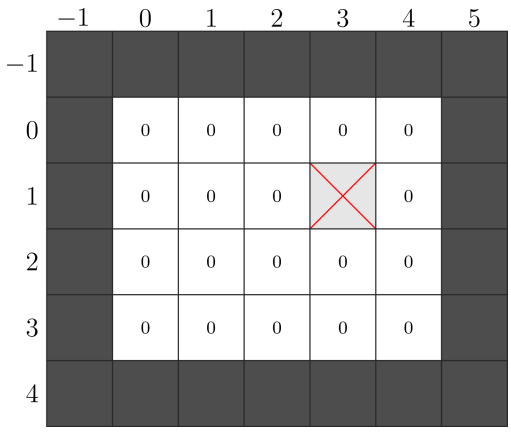
Pulibot será probado en un laberinto que consiste de una cuadrícula de celdas, que tiene $(H+2) \times (W+2)$ dimensiones. Las filas de la cuadrícula están enumeradas desde -1 hasta H de norte a sur, y las columnas están enumeradas desde -1 hasta W de oeste a este. Nos referiremos a la celda ubicada en la fila r y columna c de la cuadrícula ($-1 \leq r \leq H$, $-1 \leq c \leq W$) como celda (r, c).

Considera una celda (r, c) tal que $0 \leq r < H$ y $0 \leq c < W$. Hay 4 celdas **adyacentes** a la celda (r, c):

- La celda $(r, c-1)$ se denomina como la celda **oeste** de la celda (r, c)
- La celda $(r+1, c)$ se denomina como la celda **sur** de la celda (r, c)
- La celda $(r, c+1)$ se denomina como la celda **este** de la celda (r, c)
- La celda $(r-1, c)$ se denomina como la celda **norte** de la celda (r, c)

Se dice que la celda (r, c) es una celda **límite** del laberinto si se cumple que $r = -1$ o $r = H$ o $c = -1$ o $c = W$. Toda celda que no es un límite del laberinto es una celda **obstáculo** o una celda **vacía**. Adicionalmente, cada celda vacía tiene un **color**, representado por un número entero no negativo entre 0 y Z_{MAX}, inclusive. Inicialmente, el color de cada celda vacía es 0.

Por ejemplo, considérese un laberinto con $H = 4$ y $W = 5$, que contiene un solo obstáculo en la celda $(1, 3)$:



El único obstáculo es denotado por una cruz. Las celdas que son límites están sombreadas. El número en cada celda vacía representa su color.

Un **camino** de longitud ℓ ($\ell > 0$) desde la celda (r_0, c_0) hasta la celda (r_ℓ, c_ℓ) es una secuencia de celdas **vacías** distintas entre sí $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ en la que para cada i ($0 \leq i < \ell$), las celdas (r_i, c_i) y (r_{i+1}, c_{i+1}) son adyacentes.

Nótese que un camino de longitud ℓ contiene exactamente $\ell + 1$ celdas.

En la competencia los investigadores han instalado un laberinto en el que existe al menos un camino desde la celda $(0, 0)$ hasta la celda $(H - 1, W - 1)$. Nótese que esto implica que las celdas $(0, 0)$ y $(H - 1, W - 1)$ son vacías.

Hanga no sabe cuáles celdas del laberinto son vacías y cuáles son obstáculos.

Tu tarea es ayudar a Hanga a programar a Pulibot de tal forma que sea capaz de encontrar un *camino más corto* (esto es, un camino de longitud mínima) desde la celda $(0, 0)$ hasta la celda $(H - 1, W - 1)$ en el laberinto desconocido instalado por los investigadores. Las especificaciones de Pulibot y las reglas del concurso se describen a continuación.

Ten en cuenta que la última sección del enunciado de este problema describe una herramienta que puedes usar para visualizar a Pulibot.

Especificaciones para Pulibot

Definamos el **estado** de una celda (r, c) para cada $-1 \leq r \leq H$ y $-1 \leq c \leq W$ como un entero tal que:

- si la celda (r, c) es un límite entonces su estado es -2;
- si la celda (r, c) es un obstáculo entonces su estado es -1;
- si la celda (r, c) es una celda vacía entonces su estado es el color de la celda.

El programa de Pulibot es ejecutado como una secuencia de pasos. En cada paso Pulibot reconoce los estados de celdas cercanas y luego realiza una instrucción. La instrucción que realiza es determinada por los estados reconocidos. A continuación se muestra una descripción más precisa.

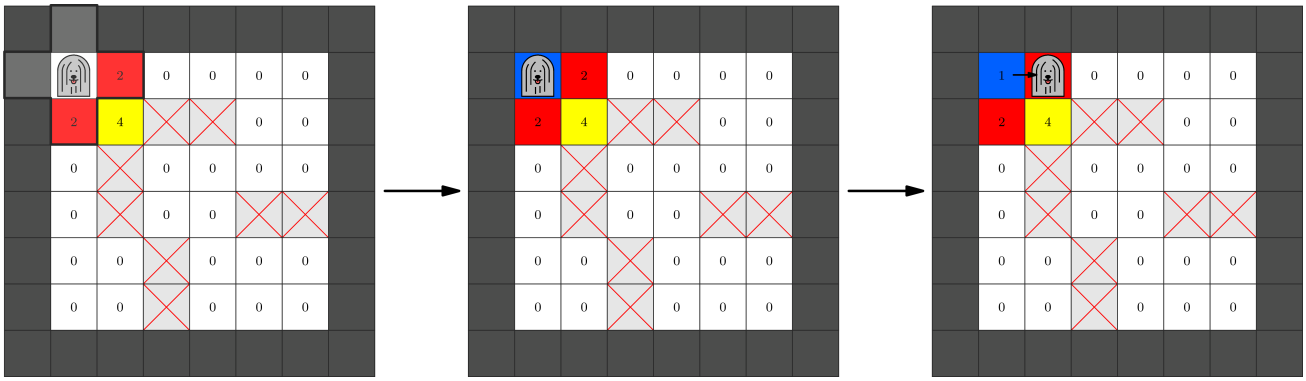
Supóngase que al principio del paso actual, Pulibot está en la celda (r, c), la cual es una celda vacía. El paso es realizado de la siguiente forma:

1. Primero, Pulibot reconoce el actual **arreglo de estado**, esto es, el arreglo $S = [S[0], S[1], S[2], S[3], S[4]]$, que consiste del estado de la celda (r, c) y el de todas las celdas adyacentes:
    - $S[0]$ es el estado de la celda (r, c).
    - $S[1]$ es el estado de la celda al oeste.
    - $S[2]$ es el estado de la celda al sur.
    - $S[3]$ es el estado de la celda al este.

- $S[4]$ es el estado de la celda al norte.

2. Luego, Pulibot determina la **instrucción** (Z, A) la cual corresponde al arreglo de estado reconocido.
3. Por último, Pulibot realiza dicha instrucción: colorea la celda (r, c) con el color Z y luego realiza la acción A, la cual es una de las siguientes acciones:
   - *Quedarse* en la celda (r, c);
   - *Moverse* a una de las 4 celdas adyacentes;
   - *Terminar el programa*.

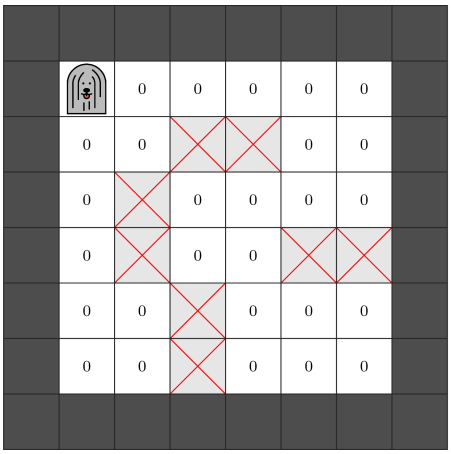
Por ejemplo, considera el escenario mostrado a la izquierda de la siguiente imagen. Pulibot está actualmente en la celda $(0, 0)$ que tiene color 0. Pulibot reconoce el arreglo de estado $S = [0, -2, 2, 2, -2]$. Pulibot podría tener un programa que, habiendo reconocido este arreglo, colorea la celda actual con el color $Z = 1$ y luego se mueve al este, como es mostrado en el medio y a la derecha de la imagen:

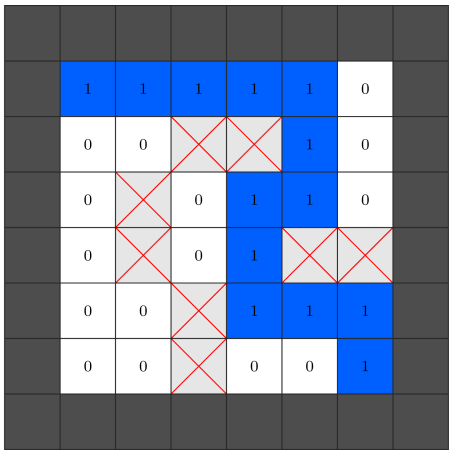


Reglas de la Competencia de Robots

- Al inicio Pulibot es colocado en la celda $(0, 0)$ y empieza a ejecutar su programa.
- Pulibot no tiene permitido moverse a una celda que no sea vacía.
- El programa de Pulibot debe terminar después de a lo sumo $500\,000$ pasos.
- Después de que el programa de Pulibot termina, las celdas vacías del laberinto deben estar coloreadas de tal forma que:
  - Exista un camino más corto desde $(0, 0)$ hasta $(H - 1, W - 1)$ para el cual el color de cada celda incluida en el camino es 1.
  - Todas las demás celdas vacías tienen color 0.
- Pulibot puede terminar su programa en cualquier celda vacía.

Por ejemplo, la siguiente imagen muestra un posible laberinto con $H = W = 6$. La configuración inicial es mostrada a la izquierda y una coloración aceptable de las celdas vacías luego de que el programa termina es mostrada a la derecha:





Detalles de Implementación

Debes implementar el siguiente procedimiento.

```
void program_pulibot()
```

- Este procedimiento debe producir el programa de Pulibot. Este programa debe funcionar correctamente para todos los valores de H y W y cualquier laberinto que cumpla con las restricciones de la tarea.
- Este procedimiento es llamado exactamente una vez por cada caso de prueba.

Este procedimiento puede hacer llamadas al siguiente procedimiento para producir el programa de Pulibot:

```
void set_instruction(int[] S, int Z, char A)
```

- S: un arreglo de longitud 5 describiendo un arreglo de estado.
- Z: un entero no negativo que representa un color.
- A: un único carácter que representa una acción para Pulibot de la forma siguiente:
  - `H`: quedarse;
  - `W`: moverse al oeste;
  - `S`: moverse al sur;
  - `E`: moverse al este;
  - `N`: moverse al norte;
  - `T`: terminar el programa.
- Llamar a este procedimiento le indica a Pulibot que, habiendo reconocido el arreglo de estado S, debe realizar la instrucción (Z, A).

Llamar a este procedimiento múltiples veces con el mismo arreglo de estado S resultará en `Output isn't correct` como veredicto.

No es requerido llamar a `set_instruction` con cada uno de los posibles arreglo de estado S. Sin embargo, si Pulibot luego reconoce un arreglo de estado para el cual no fue establecida una instrucción, obtendrás `Output isn't correct` como veredicto.

Después de que `program_pulibot` es completado, el evaluador invoca el programa de Pulibot sobre uno o más laberintos. Estas invocaciones *no* cuentan para el tiempo límite de tu solución. El evaluador *no* es adaptativo, es decir, el conjunto de laberintos está predefinido en cada caso de prueba.

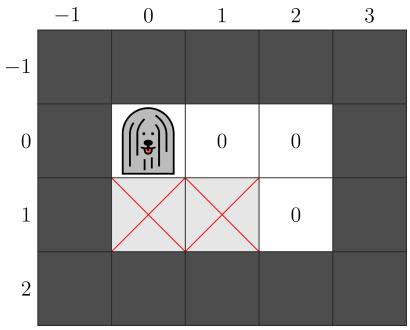
Si Pulibot viola cualquiera de las Reglas de la Competencia de Robots antes de terminar su programa, obtendrás `Output isn't correct` como veredicto.

Ejemplo

El procedimiento `program_pulibot` podría hacer llamadas a `set_instruction` como sigue:

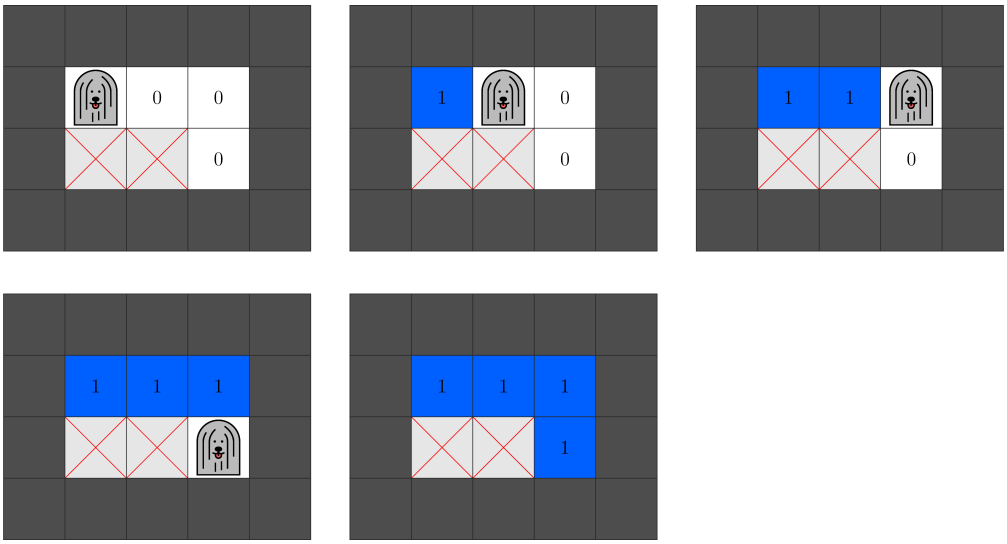
LLamada	Instrucción para el arreglo de estado S
`set_instruction([0,-2,-1,0,-2], 1, E)`	Colorear con 1 y moverse al este
`set_instruction([0,1,-1, 0,-2], 1, E)`	Colorear con 1 y moverse al este
`set_instruction([0,1,0,-2,-2],1,S)`	Colorear con 1 y moverse al sur
`set_instruction([0,-1,-2, -2,1], 1, T)`	Colorear con 1 y terminar

Considera un escenario donde $H = 2$ y $W = 3$. El laberinto es como se muestra en la siguiente imagen.



Para este laberinto en particular, el programa de Pulibot corre en cuatro pasos. Los arreglos de estado que Pulibot reconoce y las instrucciones que realiza corresponden exactamente a las cuatro llamadas a `set_instruction` hechas más arriba, en orden. La última de estas instrucciones termina el programa.

La siguiente imagen muestra el laberinto antes de cada uno de los cuatro pasos, y la coloración final luego de que el programa termina:



No obstante, debes notar que este programa de 4 instrucciones podría no encontrar un camino más corto en otros laberintos válidos. Por lo tanto, si es enviado, recibirá `Output isn't correct` como veredicto.

Restricciones

$Z_{MAX} = 19$. Por eso, Pulibot puede usar colores de 0 a 19, inclusive.

Para cada laberinto usado para probar a Pulibot:

- $2 \leq H, W \leq 15$
- Hay al menos un camino desde la celda $(0, 0)$ hasta la celda $(H - 1, W - 1)$.

Sub-tareas

1. (6 puntos) No hay obstáculos en el laberinto.
2. (10 puntos) $H = 2$
3. (18 puntos) Hay exactamente un solo camino entre cada par de celdas vacías.
4. (20 puntos) Cada camino más corto desde la celda $(0, 0)$ hasta la celda $(H - 1, W - 1)$ tiene longitud $H + W - 2$.
5. (46 puntos) Sin restricciones adicionales.

Si en cualquier caso de prueba las llamadas al procedimiento `set_instruction` o la ejecución del programa de Pulibot no cumplen con las restricciones descritas en los Detalles de Implementación, el puntaje de tu solución para esta subtarea será 0.

En cada subtarea puedes obtener un puntaje parcial produciendo una coloración que sea casi correcta.

Formalmente:

- La solución de un caso de prueba es **completa** si la coloración final de las celdas vacías satisface las Reglas de la Competencia de Robots.
- La solución de un caso de prueba es **parcial** si la coloración final luce como sigue:
  - Existe un camino más corto desde $(0, 0)$ hasta $(H - 1, W - 1)$ para el cual el color de cada celda incluida en el camino es 1.
  - No hay ninguna otra celda vacía en la cuadricula con color 1.
  - Alguna celda vacía en la cuadrícula tiene un color diferente a 0 y 1.

Si tu solución a un caso de prueba no es ni completa ni parcial, tu puntaje para dicho caso de prueba será 0.

En las subtareas 1-4, el puntaje para una solucíón completa es de 100% y el puntaje para una solución parcial a un caso de prueba es el 50% de los puntos para su subtarea.

En la subtarea 5, tu puntaje depende del número de colores usados en el programa de Pulibot. Más precisamente, denotemos por $Z^\star$ el máximo valor de Z sobre todas las llamadas hechas a `set_instruction`. El puntaje de este caso de prueba es calculado de acuerdo a la siguiente tabla:

Condición	Puntaje (completo)	Puntaje (parcial)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

El puntaje de cada subtarea es el mínimo de puntos de los casos de prueba en la subtarea.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada con el siguiente formato:

- línea 1: H W
- línea $2 + r$ ($0 \leq r < H$): $m[r][0]$ $m[r][1]$ $\ldots$ $m[r][W - 1]$

Aquí, m es un arreglo de H arreglos de W enteros, describiendo las celdas del laberinto que no son límites. $m[r][c] = 0$ si la celda (r, c) es una celda vacía y $m[r][c] = 1$ si la celda (r, c) es un obstáculo.

El evaluador de ejemplo primero llama a `program_pulibot()`. Si el evaluador detecta una violación de protocolo (protocol violation), imprimirá `Protocol Violation: <MSG>` y terminará, donde `<MSG>` es uno de los siguientes mensajes de error:

- `Invalid array`: Para algún i no se cumple que $-2 \leq S[i] \leq Z_{MAX}$, o la longitud del arreglo S no es 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ no se cumple.
- `Invalid action`: el carácter A no es ninguno de entre `H`, `W`, `S`, `E`, `N` o `T`.
- `Same state array`: `set_instruction` fue llamado con el mismo arreglo de estado S al menos dos veces.

En caso contrario, cuando `program_pulibot` completa su ejecución, el evaluador de ejemplo ejecuta el programa de Pulibot en el laberinto descrito en la entrada.

El evaluador produce dos salidas.

Primero, el evaluador escribe un registro de las acciones de Pulibot al archivo `robot.bin` en el directorio de trabajo. Este archivo sirve como la entrada de la herramienta de visualización descrita en la siguiente sección.

Segundo, si el programa de Pulibot no termina satisfactoriamente, el evaluador imprime uno de los siguientes mensajes de error:

- `Unexpected state`: Pulibot reconoció un arreglo de estado con el cual `set_instruction` no fue llamado.
- `Invalid move`: una acción realizada resultó en que Pulibot se moviera a una celda no vacía.
- `Too many steps`: Pulibot realizó $500\,000$ pasos sin terminar su programa.

De lo contrario, sea $e[r][c]$ el estado de la celda (r, c) después de que el programa de Pulibot terminó. El evaluador imprime H líneas con el siguiente formato:

- Línea $1 + r$ ($0 \leq r < H$): $e[r][0]$ $e[r][1]$ $\ldots$ $e[r][W-1]$

Herramienta de visualización

El paquete adjunto para esta tarea contiene un archivo llamado `display.py`. Cuando es invocado, este script en Python muestra las acciones de Pulibot en el laberinto descrito por la entrada del evaluador de prueba. Para esto, el archivo binario `robot.bin` debe estar presente en el directorio de trabajo.

Para invocar el script, ejecuta el siguiente comando:

```
python3 display.py
```

Una interfaz gráfica simple aparecerá. Sus funcionalidades principales son las siguientes:

- Puedes observar el estado del laberinto completo. La ubicación actual de Pulibot es resaltada por un rectángulo.

- Puedes navegar a través de los pasos de Pulibot haciendo click en los botones de flechas o presionando sus hotkeys (teclas de atajo).
- El próximo paso en el programa de Pulibot es mostrado en la parte inferior Muestra el arreglo de estado actual y la instrucción que realizará. Después del paso final, muestra o uno de los mensajes de error del evaluador, o `Terminated` si el programa termina satisfactoriamente.
- Para cada número que representa un color, puedes asignar un color de fondo visual, así como también un texto para mostrar. El texto para mostrar es una cadena de carácteres corta que aparecerá en cada celda que tenga el mismo color.
- Puedes asignar colores de fondo y textos para mostrar mediante alguna de las siguientes formas:
    - Asignándolos en una ventana de diálogo después de dar click al botón `Colors`.
    - Editando el contenido del archivo `colors.txt`.
- Para volver a cargar `robot.bin`, usa el botón de `Reload`. Es útil si el contenido de `robot.bin` ha cambiado.