Mazes (mazes)

Hampton Court Palace (en Richmond, al sudoeste de London) es famoso por su laberinto, que fue plantado por el rey William III alrededor de 1690. Ya que el palacio está hoy en día abierto al público como una atracción turística, las autoridades reales han decidido reemplazar el laberinto con una versión actualizada. Han estimado el número de turistas K que esperan que visiten durante la temporada, y les gustaría que el nuevo laberinto tuviera exactamente K rutas posibles por él, permitiendo que cada visitante tenga una experiencia única.

El laberinto es una cuadrícula de N filas y M columnas, donde cada celda puede o bien estar vacía o bien contener un seto. El laberinto está rodeado por una valla, y tiene una única entrada y una única salida. La entrada está en la celda superior izquierda, y la salida en la celda inferior derecha. El visitante debe completar el laberinto haciendo sólo movimientos hacia la derecha o hacia abajo. Debido a restricciones de espacio, el laberinto no puede tener **ni más de 200 filas ni más de 200 columnas**.

Tu objetivo es diseñar un laberinto con exactamente K rutas desde la entrada a la salida que sólo hagan movimientos hacia la derecha o hacia abajo.

Implementación

Deberás presentar un único fichero `.cpp`.

> ☞ Entre los archivos adjuntos encontrarás una plantilla `mazes.cpp` con una implementación de ejemplo.

Tienes que implementar la siguiente función:

C++
```
vector<vector<char>> solve(long long K);
```

- El entero K representa el número deseado de rutas por el laberinto.
- La función debe retornar un vector bidimensional de caracteres, representando el laberinto.
- El laberinto retornado debe tener N filas y M columnas, para algunos $N, M \leq 200$.
- Cada celda del laberinto puede ser o bien `.` (punto) para una celda vacía o bien `#` (almohadilla) para una celda con seto.
- La entrada está en la celda $(0, 0)$ y la salida está en la celda $(N-1, M-1)$.
- El laberinto debe tener exactamente K formas diferentes de ir de la entrada a la salida haciendo sólo movimientos hacia la derecha o hacia abajo.

El grader llamará a la función `solve` e imprimirá su valor de retorno en el archivo de salida.

Sample Grader

El directorio de la tarea contiene una versión simplificada del grader del jurado, que puedes utilizar para probar tu solución localmente. El grader simplificado lee los datos de entrada de `stdin`, llama a la función que debes implementar, y finalmente escribe la salida a `stdout`.

La entrada consiste en una línea, con un único entero K.

La salida consiste en varias líneas:

- La priera línea contiene dos enteros N y M $(N, M \leq 200)$, los números de filas y columnas del laberinto respectivamente.
- Las siguientes N líneas contienen M caracteres cada una, representando el laberinto.

- Cada caracter es o bien `.` (punto) para una celda vacía o bien `#` (almohadilla) para una celda con seto.

Restricciones

$1 \leq K \leq 10^{18}$.

Puntuación

Tu programa será probado en un conjunto de casos de prueba agrupados por subtarea. Para obtener la puntuación asociada a una subtarea, debes resolver correctamente todos los casos de prueba que contiene.

- **Subtask 1** [**0 puntos**]: Casos de ejemplo.
- **Subtask 2** [**9 puntos**]: $K \leq 10$.
- **Subtask 3** [**26 puntos**]: $K \leq 99$.
- **Subtask 4** [**26 puntos**]: K es una potencia de dos.
- **Subtask 5** [**39 puntos**]: Sin restricciones adicionales.

Ejemplos de entrada/salida

stdin	stdout
1	2 2 .# ..
3	8 8 ....######. ..#...#. .#..#.#. ...##... .#...##. .###.##. ....

Explicación

En el **primer caso de ejemplo**, obviamente sólo hay una forma de recorrer el laberinto.

En el **segundo caso de ejemplo**, hay tres formas de recorrer el laberinto. La celda de entrada está pintada de verde, la celda de salida en rojo, y las celdas que forman parte de los caminos en azul.

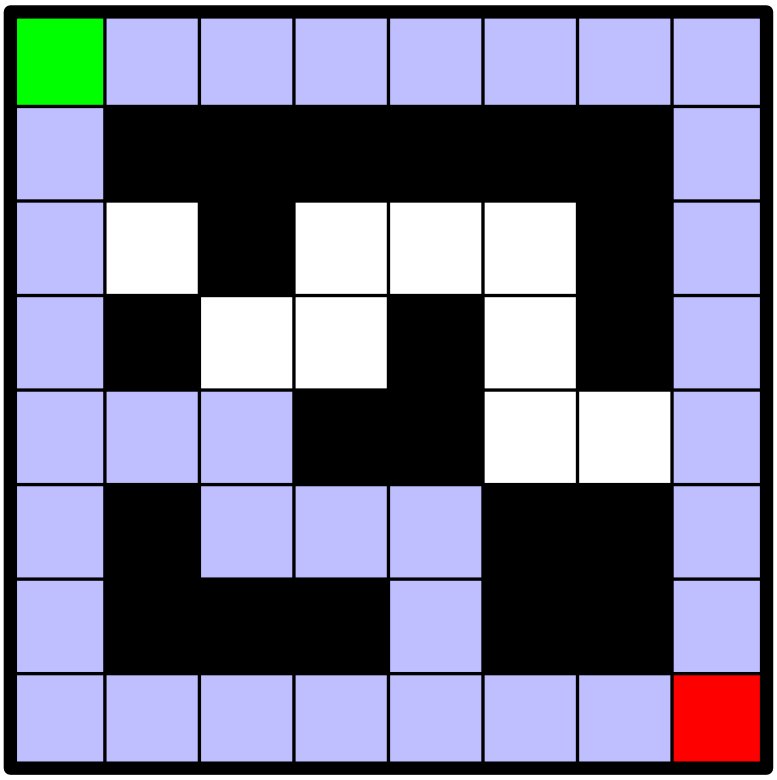

Figura 1: Segundo caso de ejemplo