Mazes (mazes)

Il palazzo della corte di Hampton (a Richmond, nel sud-ovest Londinese) è famoso per il suo labirinto, che fu piantato dal re Guglielmo III negli anni '90 del 1600. Poiché il palazzo è ora aperto al pubblico come attrazione turistica, le autorità reali hanno deciso di sostituire il labirinto con una versione aggiornata. Hanno stimato il numero di turisti K che visiteranno il labirinto durante la stagione, e vorrebbero che il nuovo labirinto abbia esattamente K percorsi possibili, permettendo a ciascun visitatore di avere un'esperienza unica.

Il labirinto è una griglia di N righe e M colonne, dove ciascuna cella può essere vuota o contenere un cespuglio. Il labirinto è circondato da una recinzione, e c'è un'unica entrata e un'unica uscita. L'entrata è nella cella in alto a sinistra, e l'uscita è nella cella in basso a destra. Il visitatore deve completare il labirinto facendo solo mosse verso destra e verso il basso. A causa di vincoli di spazio, il labirinto può avere **al massimo 200 righe** e **al massimo 200 colonne**.

Il tuo compito è progettare un labirinto con esattamente K percorsi dall'entrata all'uscita coinvolgendo solo mosse verso destra e verso il basso.

Implementazione

Dovrai inviare un singolo file sorgente `.cpp`.

> ☞ Tra gli allegati di questo task troverai un template `mazes.cpp` con un esempio di implementazione.

Dovrai implementare la seguente funzione:

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

- L'intero K rappresenta il numero desiderato di percorsi attraverso il labirinto.
- La funzione deve restituire un vettore bidimensionale di caratteri, rappresentante il labirinto.
- Il labirinto restituito deve avere N righe e M colonne, dati $N, M \leq 200$.
- Ciascuna cella del labirinto dovrebbe essere un `.` (punto) per una cella vuota o un `#` (cancelletto) per una cella contenente un cespuglio.
- L'entrata è nella cella $(0, 0)$ e l'uscita è nella cella $(N-1, M-1)$.
- Il labirinto deve avere esattamente K modi diversi per attraversarlo dall'entrata all'uscita facendo solo mosse verso destra e verso il basso.

Il grader chiamerà la funzione `solve` e stamperà il valore restituito sul file di output.

Grader di prova

La cartella del problema contiene una versione semplificata del grader, che puoi usare per testare la tua soluzione in locale. Il grader semplificato legge i dati di input da `stdin`, chiama la funzione che devi implementare, e scrive l'output su `stdout`.

L'input è formato da una singola riga contenente un singolo intero K.

L'output è formato di diverse righe, contenenti:

- La prima riga contiene due interi N e M ($N, M \leq 200$), il numero di righe e colonne del labirinto, rispettivamente.
- Le successive N righe contengono M caratteri ciascuna, rappresentanti il labirinto.

- Ciascun carattere è un `.` (punto) per una cella vuota o un `#` (cancelletto) per una cella contenente un cespuglio.

Assunzioni

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

Assegnazione del punteggio

Il tuo programma verrà testato su un insieme di test case raggruppati per subtask. Per ottenere il punteggio associato a un subtask, devi risolvere correttamente tutti i test case che contiene.

- **Subtask 1** [**0 punti**]: Casi di esempio.
- **Subtask 2** [**9 punti**]: $K \leq 10$.
- **Subtask 3** [**26 punti**]: $K \leq 99$.
- **Subtask 4** [**26 punti**]: K è una potenza di due.
- **Subtask 5** [**39 punti**]: Nessuna limitazione aggiuntiva.

Esempi di input/output

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

Spiegazione

Nel **primo caso di esempio**, c'è ovviamente un solo modo per attraversare il labirinto.

Nel **secondo caso di esempio**, ci sono tre modi possibili per attraversare il labirinto. La cella di partenza è colorata di verde, quella di arrivo di rosso, e le celle che formano i percorsi sono colorate di blu.

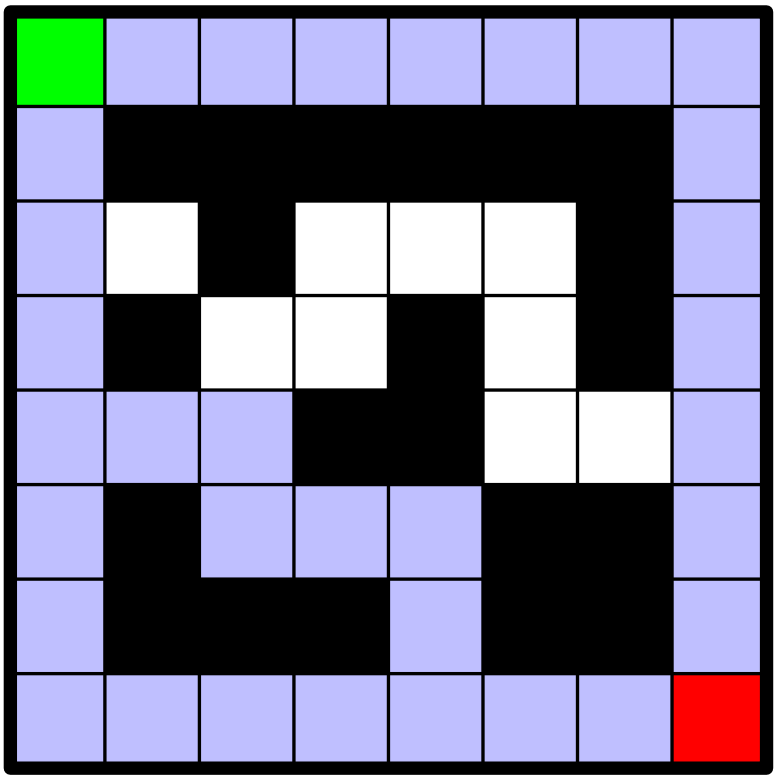

Figura 1: Secondo caso di esempio