Mazes (mazes)

Le Château de Hampton Court (situé à Richmond, au sud-ouest de Londres) est connu pour son labyrinthe, qui a été planté par le Roi William III dans les années 1690. Comme le palace est maintenant ouvert au public comme attraction touristique, les autorités royales ont décidé de remplacer le labyrinthe avec un nouveau tracé. Ils ont estimé le nombre de touristes K attendus durant cette saison, et ils voudraient que le labyrinthe ait exactement K différents chemins le traversant, permettant à chaque visiteur d'avoir une expérience unique.

Le labyrinthe est une grille de N lignes et M colonnes, où chaque cellule est soit vide, soit contient une haie. La labyrinthe est cloturé, avec une seule entrée et une seule sortie. L'entrée est à la cellule supérieure gauche, et la sortie est à la cellule inférieure droite. Les visiteurs doivent compléter le labyrinthe seulement en descendant et en allant vers la droite. Par manque d'espace, le labyrinthe **ne peut pas dépasser** les **200 lignes**, ni les **200 colonnes**.

Votre tâche est de dessiner un labyrinthe avec exactement K chemins de l'entrée à la sortie qui n'incluent que des mouvements vers le bas ou vers la droite.

Implémentation

Vous devez soumettre un seul fichier source `.cpp`.

> ☞ Parmi les pièces jointes à cette tâche, vous trouverez un modèle `mazes.cpp` avec un modèle d'implémentation.

Vous devez implémenter la fonction suivante :

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

- L'entier K représente le nombre désiré de chemins à travers le labyrinthe.
- La fonction doit renvoyer un vecteur de caractères à deux dimensions, représentant le labyrinthe.
- Le labyrinthe renvoyé doit avoir N lignes et M colonnes, avec $N, M \leq 200$.
- Chaque cellule du labyrinthe doit contenir soit un `.` (point) pour une cellule vide, soit un `#` (croisillon) pour une cellule contenant une haie.
- L'entrée se trouve dans la cellule $(0, 0)$ et la sortie dans la cellule $(N-1, M-1)$.
- Le labyrinthe doit avoir exactement K différents chemins de l'entrée à la sortie en n'allant que vers le bas ou la droite.

Le grader appellera la fonction `solve` et imprimera le résultat dans le fichier sortie.

Évaluateur

Le dossier de la tâche contient une version simplifiée du grader, qui vous permet de tester votre solution localement. Ce grader simplifié lit les données d'entrée depuis `stdin`, appelle les fonctions que vous devez implémenter, et finalement écrit le résultat dans `stdout`

L'entrée est constituée de 1 ligne, contenant l'entier K.

La sortie est constituée de plusieurs lignes, contenant :

- La première ligne contient N et M ($N, M \leq 200$), les nombre de lignes et de colonnes.
- Les N prochaines lignes contiennent M caractères, représentant le labyrinthe.

— Chaque caractère est soit un `.` (point) pour une cellule vide ou soit un `#` (croisillon) pour une cellule contenant une haie.

Contraintes

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

Score

Votre programme sera testé sur un ensemble de cas de test groupés par sous-tâche. Afin d'obtenir les points associé à une sous-tâche, vous devez répondre correctement à tout les cas de test associés.

— **Sous-tâche 1** [**0 points**]: Les exemples.

— **Sous-tâche 2** [**9 points**]: $K \leq 10$.

— **Sous-tâche 3** [**26 points**]: $K \leq 99$.

— **Sous-tâche 4** [**26 points**]: K est une puissance de 2.

— **Sous-tâche 5** [**39 points**]: Pas de contraintes additionnelles.

Exemples

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

Explication

Dans le **premier exemple**, il est évident qu'il n'y a qu'un seul chemin traversant le labyrinthe.

Dans le **deuxième exemple**, il y a trois chemins possibles pour traverser le labyrinthe. L'entrée est en vert, la sortie est en rouge, et les cellules faisant parties des chemins sont en bleu.

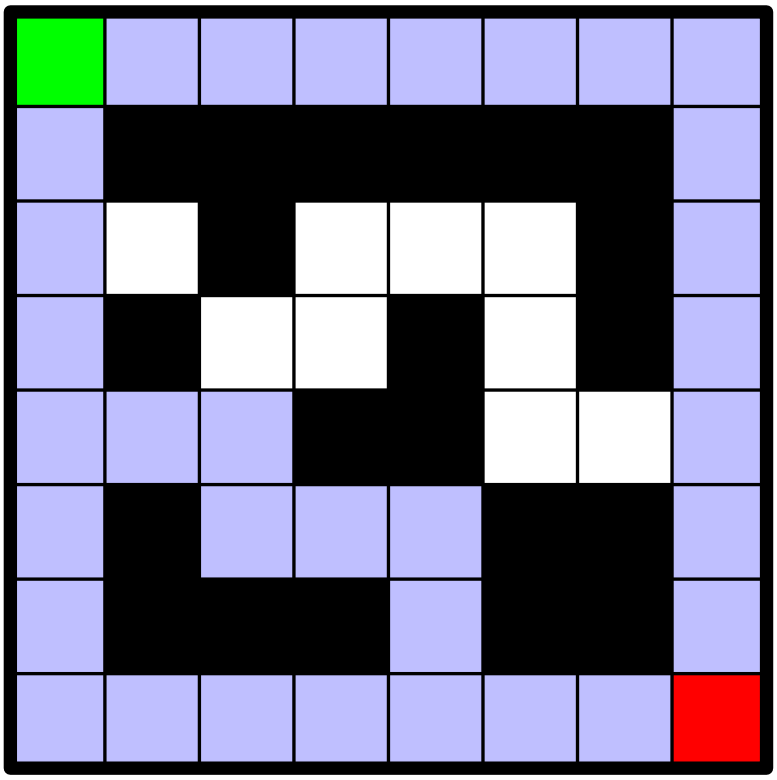

FIGURE 1 – Le deuxième exemple