Nowruz

Dans quelques jours sera fêté Nowruz (nouvel an persan), et grand-père a invité sa famille dans son jardin. Il y a k enfants parmi les invités, et grand-père a décidé d'organiser un cache-cache géant afin de les divertir.

Le jardin peut être représenté par une grille rectangulaire composée de $m \times n$ cases. Certaines cases (éventuellement aucune) sont bloquées par des rochers, les cases restantes sont dites *libres*. Deux cases sont dites *voisines* si elles ont un côté en commun. Chaque case possède donc au plus 4 cases voisines : deux dans la direction horizontale et deux dans la direction verticale. Grand-père veut transformer son jardin en un labyrinthe. Pour cela, il peut bloquer certaines cases en y plantant des buissons. Les cases sur lesquelles des buissons sont plantés ne sont alors plus libres.

Un labyrinthe doit vérifier les propriétés suivantes. Il doit exister exactement un *chemin simple* entre chaque paire a et b de cases libres. Un chemin simple entre les cases a et b est une séquence de cases libres distinctes dans laquelle la première case est a, la dernière case est b, et deux cases consécutives sont voisines.

Un enfant peut se cacher sur une case si et seulement si cette case est libre et a *exactement* une case voisine libre. Deux enfants ne peuvent pas se cacher sur la même case.

La carte du jardin vous est fournie en entrée. Votre rôle est d'aider grand-père à créer un labyrinthe dans lequel beaucoup d'enfants peuvent se cacher.

Détails d'implémentation

Il s'agit d'une tâche à score partiel de type “output-only”. Dix fichiers d'entrée vous sont fournis, chacun décrivant le jardin de grand-père. Pour chaque fichier d'entrée, vous devez soumettre un fichier de sortie avec la carte d'un labyrinthe. Pour chaque fichier de sortie, vous obtiendrez des points en fonction du nombre d'enfants pouvant se cacher dans votre labyrinthe.

Vous ne devez soumettre aucun code source pour cette tâche.

Format d'entrée

Chaque fichier d'entrée décrit une grille représentant un jardin et donne le nombre k d'enfants invités par grand-père. Le format est le suivant :

- ligne 1 : m n k
- ligne $1 + i$ (avec $1 \leq i \leq m$) : ligne i de la grille, décrite par une chaîne de taille n composée des caractères suivants (sans espace) :

- '`.`' : une case libre,
- '`#`' : un rocher.

Format de sortie

- ligne i (avec $1 \leq i \leq m$) : ligne i du labyrinthe (jardin après que les buissons ont été plantés), décrite par une chaîne de taille n composée des caractères suivants (sans espace) :
  - '`.`' : une case libre,
  - '`#`' : un rocher,
  - '`X`' : un buisson. (La lettre X doit être en majuscule.)

Contraintes

- $1 \leq m, n \leq 1024$

Score

Un fichier de sortie est *valide* s'il satisfait aux conditions suivantes :

- La carte de sortie doit correspondre à la carte d'entrée, à l'exception d'un nombre arbitraire de caractères '`.`' qui peuvent avoir été transformés en '`X`' (cases bloquées par les buissons).
- La carte de sortie doit vérifier les propriétés d'un labyrinthe, comme définies dans l'énoncé.

Si l'une de vos sorties n'est pas valide, votre score pour ce test sera 0. Sinon votre score sera de $\min(10, 10 \cdot l/k)$ points, tronqué à deux chiffres après la virgule, où l est le nombre d'enfants pouvant se cacher dans votre labyrinthe et k est le nombre fourni dans le fichier d'entrée. Vous marquerez 10 points pour un test si et seulement si votre sortie est un labyrinthe dans lequel au moins k enfants peuvent se cacher. Pour chaque test il existe une sortie qui obtient 10 points.

Notez que si votre solution est valide mais obtient 0 point avec la formule ci-dessus, le jugement qui apparaîtra sur l'interface en ligne (CMS) sera 'Wrong Answer'.

Exemple

Sur l'entrée suivante :

```
4 5 5
....#
.#..#
...#.
....#
```

Une sortie valide est :

```
.X.X#
.#..#
...#X
XX..#
```

Comme $l = 4$ enfants peuvent se cacher dans ce labyrinthe, cette solution obtiendra $10 \cdot 4/5 = 8$ points. Les cases sur lesquelles des enfants peuvent se cacher sont représentées avec un `O` ci-dessous :

```
OXOX#
.#.O#
...#X
XX.O#
```

Les trois sorties suivantes ne sont pas valides :

```
.XXX#        ...X#        XXXX#
.#XX#        .#.X#        X#XX#
...#.        ...#X        ..X#X
XX..#        XXXX#        ..XX#
```

Dans la sortie de gauche, il n'existe pas de chemin simple entre la case en haut à gauche et la case libre de la colonne de droite. Dans les deux autres sorties il existe exactement deux chemins simples pour chaque paire de cases libres distinctes.