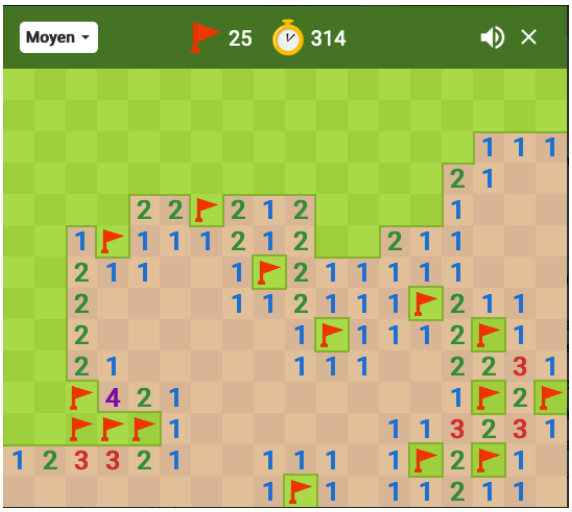
Université de Sherbrooke
Département d'informatique

Le démineur



Description du problème

Vous devez écrire une partie du jeu Démineur. Vous devrez coder la partie du jeu qui découvre toutes les cases vides adjacentes à une case donnée (méthode `essaieCase`). L'algorithme à utiliser **devra être récursif**.

Le jeu de Démineur utilise une carte sur laquelle certaines cases sont des mines. Le but est de découvrir toutes les cases qui ne sont pas des mines. Si le joueur découvre une case qui est une mine, la partie est perdue. La partie démarre donc avec toutes les cases fermées.

Le joueur choisi une ligne et une colonne et le programme ouvre la case. Si c'est une mine, la partie est perdue. Si c'est une case vide, le programme doit ouvrir automatiquement toutes les cases adjacentes jusqu'à ce qu'une de ces cases indique qu'une case adjacente est une mine (peu importe laquelle). Le joueur peut rechoisir une autre position dans la grille et ouvrir d'autres cases. Le jeu se termine lorsque les seules cases restantes à ouvrir sont les mines.

Vous **devez** utiliser la base de code fournie sur le site web pour ce devoir. Vous devez aussi compléter le code de deux autres méthodes de la classe `carte`. **Vous ne devez pas modifier les modules déjà codés**.

De plus, vous devez concevoir une classe `Position` qui contiendrait les numéros de la ligne et de la colonne d'une position dans la grille. Voyez le code fourni pour concevoir votre classe.

1. La carte

La carte doit être lue à partir d'un fichier. Votre programme doit donc demander un nom de fichier pour la carte au début de l'exécution. Le format du fichier de carte est le suivant :

- Un entier N, représentant le nombre de colonne à lire ;
- Un entier M, représentant le nombre de ligne ;
- M séquences de caractères (0 ou 1), de longueur N, séparés par un espace.

Les entrées du fichier (0 ou 1) pour la carte représentent soit une case vide (0), soit une mine (1).

Voici un exemple de fichier représentant une carte 10×15:

```
10
15
0001011001
1000001000
0000000000
0010000000
0000000001
0000000000
0000000000
0000000000
0000100000
0000100000
0001000000
0010100000
0001000000
0000000100
0000000000
```

Vous devrez générer vous-même les informations sur le nombre de mines adjacentes à une case. Vous devrez donc, après avoir lu la carte, indiquer sur celle-ci les cases qui doivent contenir des nombres à afficher.

L'opérateur d'affichage de votre carte doit s'occuper d'afficher le bon nombre lorsque la case comporte une chiffre.

2. Ouvrir les cases vides

La solution la plus simple et la plus élégante pour ouvrir toutes les cases vides adjacentes à une cases vides, jusqu'aux cases contenant des chiffres, est d'utiliser un algorithme récursif. La fonction doit retourner vrai si la case a pu être ouverte sans problème. Elle retourne faux si on a ouvert une mine.

fonction essaieCase

```
Entrées
  position (ligne et colonne) :
Sortie
    booléen : vrai si la case est ouverte, faux sinon
```

```
si la position est déjà ouverte
   retourner vrai
fin_si
ouvrir la case courante
si la case est une mine
   retourner faux
fin_si
si la case n'est pas une mine ou un chiffre
   pour_toutes les 8 cases adjacentes
      essaieCase(positionAdjacente)
   fin_pour
fin_si
retourner vrai
```

Attention! Cet algorithme est très général. Il est possible que vous ayez à le modifier