EISTI CERGY - PAU	**Classe préparatoire 1ère année** **Labyrinthe** *S.* Bornhofen, *F.* Devin, *B.* George, *P.* Loubière, *J.* Mercadal,	
	Matière : **Informatique**	*Échéance* : **Mars 2014**
		Nombre de pages du sujet : **8**

1 Introduction

L'objectif de ce mini-projet est la création aléatoire d'un labyrinthe, ainsi que sa résolution. Il existe de nombreuses techniques pour créer des labyrinthes automatiquement. Dans ce projet, vous veillerez à documenter chacune de vos fonctions. Chaque fonction ne devra en aucun cas dépasser plus de 25 lignes.

2 Création d'un labyrinthe

Voici un exemple du déroulement de l'algorithme que vous devez employer. Supposons que l'utilisateur demande 3 cases en largeur, et 2 cases en hauteur.

2.1 Initialisation

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 1 & -1 & 2 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 3 & -1 & 4 & -1 & 5 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

Quelques questions à se poser :
- Comment créer la matrice à la bonne taille ?
- Comment demander la saisie à l'utilisateur ?

2.2 Étape 1

Sélection aléatoire d'un mur, effacement de celui-ci, puis propagation de la plus petite valeur :

étape 1a :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 1 & -1 & 2 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 3 & -1 & 4 & -1 & 5 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

étape 1b :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 1 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 4 & -1 & 5 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

Quelques questions à se poser :
- Comment sélectionner un mur de manière efficace ?
- Peut on sélectionner n'importe quel mur ?
- Comment propager la plus petite valeur de manière efficace ?

2.3 Étape $2 \rightarrow n$

Même principe que l'étape 1 :

étape 2a :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 1 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 4 & -1 & 5 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

étape 2b :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 1 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 4 & 4 & 4 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

étape 3a :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 1 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 4 & 4 & 4 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

étape 3b :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 1 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 & 2 & 2 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

étape 4a :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 1 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 & 2 & 2 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

étape 4b :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 & 2 & 2 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

étape 5a :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 & 2 & 2 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

étape 5b :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

résultat :
$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

Ce qui donne :

```
* * * * * * *
*       *   *
*   *   *   *
*   *       *
* * * * * * *
```

Quelques questions à se poser :
- Comment détecter la fin de l'algorithme ?
- Comment afficher un labyrinthe ?

Une fois ceci fait vous passerez à la résolution du labyrinthe. C'est à dire positionner un point de départ (A) et un point d'arrivée (B). Puis vous afficherez le chemin le plus court pour aller de A à B.

3 Recherche du plus court chemin

Prenons l'exemple suivant, supposons que l'on ait la matrice suivante (représentant un labyrinthe) :

```
* B * * * * * * * * * * *
*       *           *   *
* * *   * * * * *   *   *
*                       *
*   * * * * *   *   * * *
*   *           *       *
*   * * *   * * *   * * *
*       *       *   *   *
* * *   * * *   *   *   *
*   *       *   *       *
*   * * *   *   * * *   *
*           *       *   *
*   *   * * *   *   * * *
*   *   *       *       A
* * * * * * * * * * * * *
```

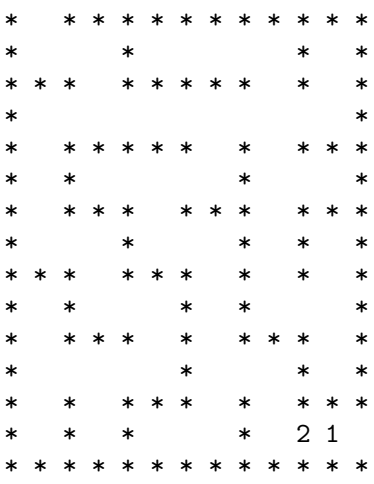
Pour trouver le plus court chemin, il suffit de numéroter les cases dans leur ordre d'adjacence en partant du départ (A) avec comme valeur initiale 1.

3.1 Début de l'algorithme

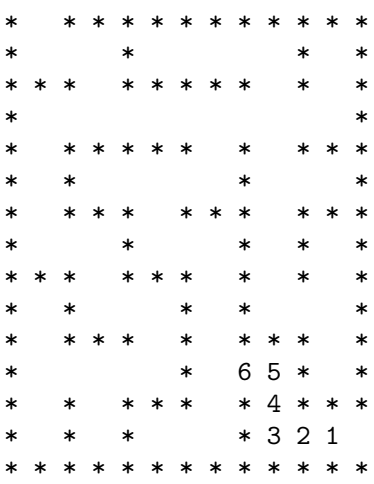
Étape 1 :

```
*   * * * * * * * * * * *
*       *           *   *
* * *   * * * * *   *   *
*                       *
*   * * * * *   *   * * *
*   *           *       *
*   * * *   * * *   * * *
*       *       *   *   *
* * *   * * *   *   *   *
*   *       *   *       *
*   * * *   *   * * *   *
*           *       *   *
*   *   * * *   *   * * *
*   *   *       *     1
* * * * * * * * * * * * *
```

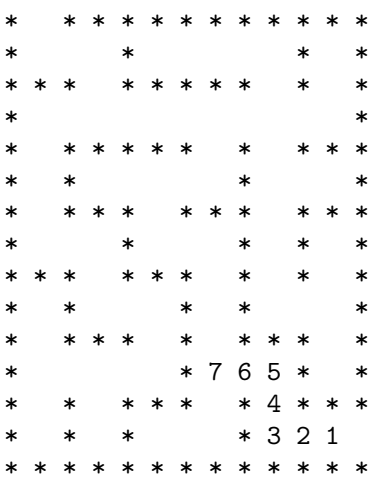
Étape 2 :



Étape 6 :



Étape 7 :



Étape 9 :

```
*   * * * * * * * * * * *
*       *           *   *
* * *   * * * * *   *   *
*                       *
*   * * * * *   *   * * *
*   *           *       *
*   * * *   * * *   * * *
*       *       *   *   *
* * *   * * *   *   *   *
*   *       * 9 *       *
*   * * *   * 8 * * *   *
*           * 7 6 5 *   *
*   *   * * * 8 * 4 * * *
*   *   *     9 * 3 2 1
* * * * * * * * * * * * *
```

Étape 17 :

```
*   * * * * * * * * * * *
*       *           *   *
* * *   * * * * *   *   *
*                       *
*   * * * * *   *   * * *
*   * H G F G H *       *
*   * * * E * * *   * * *
*       * D C B *   *   *
* * *   * * * A *   *   *
*   *       * 9 *       *
*   * * *   * 8 * * *   *
*           * 7 6 5 *   *
*   *   * * * 8 * 4 * * *
*   *   * B A 9 * 3 2 1
* * * * * * * * * * * * *
```

Ce qui donne au final (les lettres sont utilisées comme des nombres supplémentaires) :

```
*   * * * * * * * * * * *
* R Q P * R Q P O N * P *
* * * O * * * * * M * O *
* P O N M L K J K L M N *
* Q * * * * * I * M * * *
* R * H G F G H * N O P *
* S * * * E * * * O * * *
* T U V * D C B * P * V *
* * * W * * * A * Q * U *
* h * X Y Z * 9 * R S T *
* g * * * a * 8 * * * U *
* f e d c b * 7 6 5 * V *
* g * e * * * 8 * 4 * * *
* h * f * B A 9 * 3 2 1
* * * * * * * * * * * * *
```

Une fois cette matrice complétée, il suffit de partir de l'arrivée et suivre les valeurs décroissantes pour arriver au départ. On obtient alors le plus court chemin dans le labyrinthe. Ici le plus court chemin est :

```
*   * * * * * * * * * * *
* R Q P *           *   *
* * * O * * * * * *   *   *
*     N M L K J         *
*   * * * * * I *   * * *
*   *     F G H *       *
*   * * * E * * *   * * *
*       * D C B *   *   *
* * *   * * * A *   *   *
*   *       * 9 *       *
*   * * *   * 8 * * *   *
*           * 7 6 5 *   *
*   *   * * *   * 4 * * *
*   *   *       * 3 2 1
* * * * * * * * * * * * *
```

Quelques questions à se poser :

- Peut on choisir n'importe quel mur pour débuter/terminer l'algorithme ?
- Comment numéroter les cases ?
- Comment obtenir le plus court chemin, en particulier lorsqu'il existe plusieurs plus court chemins ?

4 Exemple d'exécution de votre programme

Votre programme devra avoir exactement la sortie suivante (bien évidemment les valeurs sont des exemples, et le labyrinthe est un labyrinthe généré aléatoirement) :

```
./labyrinthe
Entrer la taille de votre labyrinthe :
Largeur : 6
Hauteur : 7
Voici le labyrinthe aléatoire généré :
* * * * * * * * * * * *
*       *           *   *
* * *   * * * * *   *   *
*                       *
*   * * * * *   *   * * *
*   *           *       *
*   * * *   * * *   * * *
*       *       *   *   *
* * *   * * *   *   *   *
*   *       *   *       *
*   * * *   *   * * *   *
*           *       *   *
*   *   * * *   *   * * *
*   *   *       *       *
* * * * * * * * * * * * *
Entrer les coordonnées du point de départ :
X : 1
Y : 0
Entrer les coordonnées du point d'arrivée :
X : 12
Y : 13
Voici le plus court chemin :
```

```
* A * * * * * * * * * * *
* . . . *           *   *
* * * . * * * * * *   *   *
*     . . . . . .         *
*   * * * * * . *   * * *
*   *     . . . *       *
*   * * * . * * *   * * *
*       * . . . *   *   *
* * *   * * * . *   *   *
*   *       * . *       *
*   * * *   * . * * *   *
*           * . . . *   *
*   *   * * *   * . * * *
*   *   *       * . . . B
* * * * * * * * * * * * *
```

5 Pour aller plus loin

Pour répondre aux questions suivantes, vous créerez d'autres programmes, même s'ils peuvent s'appuyer sur la même base. Donc à chaque question correspond un programme différent. Ce programme pour le nom : `labyrinthe`*section*

5.1 Plus court chemin

Le problème de l'algorithme du plus court chemin précédent est le fait que l'on parcourt toutes les cases pour trouver la solution. Or nous aurions pu nous arrêter bien avant, comme le montre le labyrinthe ci-dessous :

```
*   * * * * * * * * * * *
* R Q P * R Q P O N * P *
* * * O * * * * * M * O *
* P O N M L K J K L M N *
* Q * * * * * I * M * * *
* R * H G F G H * N O P *
*   * * * E * * * O * * *
*       * D C B * P *   *
* * *   * * * A * Q *   *
*   *       * 9 * R     *
*   * * *   * 8 * * *   *
*           * 7 6 5 *   *
*   *   * * * 8 * 4 * * *
*   *   * B A 9 * 3 2 1
* * * * * * * * * * * * *
```

– Comment réaliser cette optimisation ?

5.2 Plus court chemin 2

Il existe une autre façon de trouver le plus court chemin. C'est de faire un parcours en main droite. Le parcours en main droite consiste à simuler le fait que l'on pose la main droite sur le mur, et que l'on suive ce mur. L'étape 9 de l'algorithme précédent devient alors :

```
*   * * * * * * * * * * *
*       *           *   *
* * *   * * * * *   *   *
*                       *
*   * * * * *   *   * * *
*   *           *       *
*   * * *   * * *   * * *
*       *       *   *   *
* * *   * * *   *   *   *
*   *       * 9 *       *
*   * * *   * 8 * * *   *
*           * 7 6 5 *   *
*   *   * * *   * 4 * * *
*   *   *       * 3 2 1
* * * * * * * * * * * * *
```

- Implémenter cet algorithme.
- Comparer le nombre de cases explorées avec cette version, et la version améliorée dans la partie 5.1. Vous afficherez donc le nombre de cases explorées pour chaque algorithme en plus du labyrinthe lui-même.

5.3 Troisième dimension

Les labyrinthes que vous avez générés précédemment sont en deux dimensions. Nous pouvons considérer une troisième dimension : l'élévation. Il faut alors placer des escaliers pour monter et descendre.

- Comment étendre la génération pour qu'elle puisse générer des labyrinthes en 3 dimensions ?
- Comment adapter les fonctions de recherche du plus court chemin ?
- Comment afficher correctement ce type de labyrinthe ?

5.4 Cycles

Lors de la génération du labyrinthe, nous nous arrêtions lorsque le labyrinthe était généré. Nous pouvons aussi envisager de continuer à enlever x murs supplémentaires.

- Générer un tel labyrinthe.
- Adapter la méthode initiale du plus court chemin pour ce nouveau type de labyrinthe.
- Adapter la méthode de la main droite pour ce nouveau type de labyrinthe.