Introduction à C/C++
TD6: Allocation dynamique.

Vincent GAUDILLIERE. vincent.gaudilliere@inria.fr.

1 Jeu de la vie.

Le jeu de la vie est un automate cellulaire qui fait évoluer une grille à deux dimensions dont les cases peuvent prendre deux valeurs possibles, *vivante* (1) ou *morte* (0). L'évolution est définie par une succession de générations. Une cellule est vivante à la génération i si et seulement si l'une des deux propriétés suivantes est vérifiée :

1. la cellule était vivante et avait exactement deux voisines vivantes à la génération $i-1$, **ou**
2. la cellule avait exactement trois voisines vivantes à la génération $i-1$.

On choisit de représenter le Jeu de la vie par une structure définie de la façon suivante :

```
typedef struct{
    char ** grille; // grille
    int taille; // taille de la grille
    int gen; // génération
}JeuDeLaVie;
```

Question 0. Ecrire un programme qui contient la définition de la structure précédente, ainsi qu'un `main` dans lequel une variable `Partie` de type `JeuDeLaVie` est déclarée. L'élément `grille` doit être initialisé à `NULL`, et les éléments `taille` et `gen` à 0.

Les questions suivantes demandent de créer des fonctions qui devront agir sur la variable `Partie`. Ces fonctions devront donc être appelées dans le `main` (après la déclaration de `Partie`, et dans l'ordre dans lequel elles apparaissent dans l'énoncé).

Question 1. Ecrire une fonction `void saisirTaille( JeuDeLaVie * J )` qui demande à l'utilisateur d'entrer une taille $n \leq 50$ pour la grille, et la stocke dans l'élément `taille` de la variable pointée par J.

Question 2. Ecrire une fonction `int creerGrille( JeuDeLaVie * J )` qui crée une grille de taille $n \times n$ en allouant (dynamiquement) l'espace nécessaire. Cette fonction doit retourner 0 si l'allocation s'est bien passée, 1 sinon. Après appel de cette fonction dans le `main`, il doit donc être possible d'accéder aux différentes cases de `Partie.grille` sans provoquer de faute de segmentation.

Les fonctions suivantes (Questions 3 à 7) ne doivent être appelées que si l'allocation de mémoire précédente s'est déroulée correctement. Si ce n'est pas le cas, le programme doit simplement afficher un message d'erreur. De plus, pour respecter les bonnes pratiques vues en cours, il convient de traiter la question 7 avant les autres.

Question 3. Ecrire une fonction `void initGrille( JeuDeLaVie * J )` qui initialise au hasard les éléments de la grille.

Question 4. Ecrire une fonction `void afficheGrille( JeuDeLaVie * J )` qui affiche la grille à l'écran. Cette fonction doit également indiquer à l'utilisateur quelle est la génération qu'il observe.

La fonction suivante (Question 5) doit être appelée 100 fois consécutivement dans le `main`. Il pourra être utile de coupler l'appel de cette fonction avec l'appel de la fonction d'affichage de la grille (Question 4), afin d'observer l'évolution de cette dernière.

Question 5. Ecrire une fonction `void nvlleGeneration( JeuDeLaVie * J )` qui met à jour l'état de la grille selon les règles présentées en début d'énoncé, et incrémente l'élément `gen` de la variable pointée par J.

Question 6. Ecrire une fonction `int sauvegarder( JeuDeLaVie * J )` qui écrit dans un fichier `Partie.txt` les éléments de la variable pointée par J. Cette fonction doit retourner 0 si l'ouverture du fichier s'est bien passée, 1 sinon. Plus précisément, la première ligne du fichier doit contenir la taille de la grille, la deuxième ligne doit contenir la génération actuelle, et la troisième ligne doit contenir les valeurs de la grille séparées par un espace, dans l'ordre suivant :

`grille[0][0] grille[0][1] ... grille[0][n-1] grille[1][0] ... .`

Question 7. Ecrire une fonction `void supprGrille( JeuDeLaVie * J )` qui libère la mémoire allouée à la Question 2.