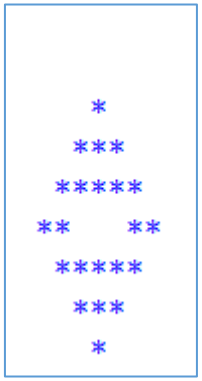
GAME Of LIFE – Le jeu de la vie

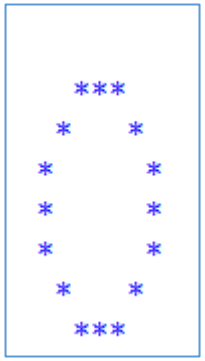
Le Jeu de la Vie est un automate cellulaire qui se déroule sur une grille à deux dimensions.

https://fr.wikipedia.org/wiki/Jeu_de_la_vie

Une génération est représentée par un tableau de caractères qui sont soit des '*' quand il s'agit d'êtres vivants, soit des caractères ' ' (espace) quand il n'y a pas d'être en vie à ces endroits.



Génération G



Génération G+1 (avec règles légèrement différentes)

La génération suivante est déterminée en appliquant les règles suivantes :

- Pour qu'une vie apparaisse dans une case, il faut qu'elle ait trois êtres vivants comme voisins.
- Pour qu'une vie subsiste dans une case, il faut qu'elle ait deux ou trois êtres vivants comme voisins.
- Dans les autres cas, il n'y aura pas de naissance ou la vie disparaitra pour cause d'isolement ou de surpopulation.

a. Question : Comment va-t-on gérer les générations successives ? De combien de tableaux a-t-on besoin ? est-il nécessaire de faire des recopies de générations d'un tableau vers l'autre ?

b. Donner le **pseudocode** de l'algorithme qui, à partir d'un nombre de voisins compris entre 0 et 8, et l'état actuel d'une cellule ('en vie' ou 'mort'), détermine l'état de cette cellule à la prochaine génération en appliquant les règles précédentes.

c. Donner le **pseudocode** de la fonction qui implémente l'algorithme de présence de vies à la génération suivante.
Celle-ci renvoie le caractère **'*'** quand une vie doit être placée ou **' '** (espace) si elle n'existe pas compte tenu du nombre de ses voisins et de son état actuel.

d. Réaliser une implémentation complète du Jeu de la Vie, en fournissant le code des fonctions listées dans le fichier GameOfLife.h.
Ecrire le projet du jeu de la vie en utilisant la syntaxe sans crochet pour le passage des tableaux en paramètre de fonction et dans le corps de la fonction elle-même.

Remarque : Sous Visual Studio, utiliser les fichiers ConsoleTools.h et ConsoleTools.c pour afficher les caractères * et espace à l'écran aux endroits désirés.

Exemple: Pour afficher une * en position 10,10 :

```
#include "ConsoleTools.h"
int main() {
      openConsole();
      moveCursor(10,10);
      plotChar('*');
      moveCursor(15,0); // ramène le curseur en colonne 0
      closeConsole();
      return 0;
      }
```

GameOfLife.h

```c
#define LifeSizeX 30
#define LifeSizeY 30
#define NBLIVES 150
#define Alive '*'
#define Dead ' '

// Gen1 et Gen2 contiennent les générations successives d'êtres en vie ou morts
char *Gen1; // Gen1 et Gen2 doivent être initialisés avec une allocation dynamique
char *Gen2;

// initLife initialise un tableau, d'abord complètement avec des morts puis place les êtres en vie et renvoie leur nombre
int initLife(char *life, int sizeX, int sizeY);

// nextGen calcule l'apparition ou la disparition des êtres à la génération suivante, de Gen1 vers Gen2 et renvoie leur nb
int nextGen(char *gen1, char *gen2, int sizeX, int sizeY);

// displayGen Affiche une génération sur la console
void displayGen(char *life, int sizeX, int sizeY);

// countNeighbours dénombre les êtres en vie autour de la postion (posX, posY) dans le tableau Life (une génération)
int countNeighbours(char *life, int sizeX, int sizeY, int posX, int posY);

// Détermine s'il y a une naissance ou une disparition en fonction du nombre de voisins et de l'état actuel
char lifeState(int nbNeighbours, char state);
```

GameOfLife.c (extrait)

```c
// calcul de la génération suivante
int nextGen(char *gen1, char *gen2, int sizeX, int sizeY) {
	int nbLives = 0;
	// on crée dans gen2 les êtres vivants à partir de l'état du tableau Gen1
	for (int l = 0; l < SizeY; l++) { // les variables l et c représentent la position d'une case en (ligne, colonne)
		for (int c = 0; c < SizeX; c++) {
			*(gen2+(l*sizeX + c)) = lifeState(countNeighbours(gen1,sizeX,sizeY,c,l), *(Gen2+(l*sizeX + c)));
			if (*(gen2+(l*sizeX + c))==Alive) nbLives++;
		}
	}


	return(nbLives);
};
```