Objectif : créer deux bibliothèques de fonctions sur les graphes : l'une avec la représentation des graphes par listes d'adjacence, l'autre par matrice d'adjacence.

Testez les fonctions au fur et à mesure via un programme principal.

Avec liste d'adjacence

On considèrera dans cette partie des graphes orientés pouvant être valués. On définit donc le triplet (source, valeur, destination) pour chaque arête.

Question 1 *Pour cette partie, vous devez commencer par reprendre (finir) les fonctions sur les listes. Vous aurez besoin du type liste contenant une valeur (float), une destination (int) et un suivant ainsi que des fonctions ci-dessous :*

```
 Liste ListeVide();
int EstListeVide(Liste);
float getValuation(Liste);
// retourne la valuation du premier maillon d'une liste non vide
int getDestination(Liste);
 // retourne la destination du premier maillon d'une liste non vide
Liste getSuivant(Liste);
 // retourne le suivant du premier maillon d'une liste non vide
Liste ajoutEnTete(Liste, float, int); // retourne la liste avec un nouveau maillon en tete
                                      // de valuation v et de destination d
```

Question 2 *Dans un fichier grapheL.h, créer un type sommet défini par une étiquette et une liste de successeurs. Créer le type grapheL défini comme un nombre de sommets et un tableau de sommets.*

Avant de programmer, faire un schéma représentant le type grapheL.

Question 3 *Implanter les fonctions suivantes dans le grapheL.c correspondant :*

```
int estVideGrapheL (GrapheL); // teste si un graphe est vide
Sommet nouveauSommet(char  etiq); // cree le sommet s d'étiquette etiq
GrapheL nouveauGrapheL(int); // cree un graphe a n sommets
int getNbSommets(GrapheL); // retourne le nombre de sommets d'un graphe non vide
Sommet getSommet(GrapheL g, int k); // retourne le k-ieme sommet d'un graphe non vide
Liste getSuccesseurs(Sommet); // retourne la liste des successeurs du sommet s
void nouvelArc(GrapheL g, int s, int d, float v); // ajoute un arc (s,v,d) au graphe non vide g
void putEtiquette(GrapheL g, int k, char* e ); // change l'etiquette du k-ieme sommet de g
char getEtiquette(GrapheL g, int k); // retourne l'etiquette du k-ieme sommet de g
```

Question 4 *Utiliser ces fonctions dans un programme principal pour initialiser un graphe et l'afficher. On affichera la liste des arêtes sous la forme (source, valuation, destination). (utiliser un graphe vu en cours)*

Les parcours

Question 5 *Implanter les algos vus en cours : visiteEnProfondeur et parcoursEnProfondeur. Ajouter un affichage à chaque entrée dans une fonction. Tester le parcours sur l'exemple vu en cours, vérifier l'appel des fonctions.*

```
 void visiteEnProfondeur(Graphe g, int s, int* marque);
void parcoursEnProfondeur(Graphe G);
```

Question 6 *Implanter le parcours en largeur. Pour celui-ci vous devrez utiliser une file. Reprenez donc (ou finissez) votre implantation sur le type file.*

Question 7 *Implanter les algos vus en cours : visiteEnLargeur et parcoursEnLargeur. Tester le parcours sur l'exemple vu en cours.*

Avec matrice d'adjacence

Question 8 *Créer un fichier grapheM.h avec une structure de graphe utilisant une matrice et contenant le nombre de sommets. On considèrera que les sommets sont nommés par des entiers. Dans la matrice, on trouvera 0 si l'arc n'existe pas et son poids sinon. (des 1 pour un graphe non pondéré.) Un graphe sera donc un pointeur vers cette structure.*

Question 9 *Implanter les fonctions suivantes dans le grapheM.c correspondant :*

```
int estVideGrapheM (GrapheM ); // teste si un graphe est vide
GrapheM NouveauGrapheM (int); // cree un graphe a n sommets
int getNbSommets_M(GrapheM ); // retourne le nombre de sommets d'un graphe non vide
int* getSuccesseurs_M (GrapheM , Sommet); // retourne la ligne des successeurs du sommet s
int getArc(GrapheM, Sommet, Sommet); //retourne le poids de l'arc du sommet 1 au sommet 2

void nouvelArc_M (GrapheM, Sommet , Sommet , int);
 // ajoute un arc (s,v,d) au graphe non vide g
```

Question 10 *Utiliser ces fonctions dans un programme principal pour initialiser un graphe et l'afficher. (utiliser un graphe vu en cours)*