Exercice 1. On donne les types

```c
#define MAX_VERTICES 10
typedef int edge_val ;// étiquette sur les arcs
typedef int vertex_name_t ;//nom des sommet

typedef struct edge_list_t{//liste de voisins = extrémités d'arcs
  int dest ;// extrémité de l'arc
  edge_val data ; //étiquette de l'arc
  edge_list_t * next ;
} edge_list_t ;

typedef struct vertex_t {
  vertex_name_t name ;// nom du sommet ; origine des arcs issus de ce sommet)
  struct edge_list_t * edges ;// liste de voisins
}vertex_t ;

typedef struct graph_t {
  int nb_vertices ;
  struct vertex_t *vertices[ MAX_VERTICES ] ;
} graph ;
```

Question 1.

Écrire la fonction `graph * create(int n);` qui crée un graphe à n sommets sans arête.

Question 2.

Écrire la fonction `void add_edge(graph * g, int i, int j);` qui ajoute j comme voisin de i. On demande une complexité $O(1)$: j est supposé absent des voisins de i.

Question 3.

Écrire la fonction `bool neighbor(graph* g, int i, int j);` qui teste si j est voisin de i.

Question 4.

Écrire la fonction `int deg(graph* g, int i);` qui renvoie le degré sortant de i.

Question 5.

Écrire la fonction `void display(graph* g);` qui affiche le graphe sous la forme [[1,5],...,[5,2,3]] (liste des voisins de chaque sommet).

Avec

```c
int main(void){

  graph * g = create(4) ;
  add_edge(g,0,1) ;add_edge(g,1,1) ;add_edge(g,1,2) ;
  add_edge(g,2,0) ;add_edge(g,2,1) ;add_edge(g,3,1) ;

  display(g) ;

  int i=1,j=0 ;
  printf("%d voisin de %d  : %d\n",i,j,neighbor(g,j,i)) ;
  printf("%d voisin de %d  : %d\n",j,i,neighbor(g,i,j)) ;
  printf("deg(%d)=%d\n",1,deg(g,1)) ;
 }
```

On obtient l'affichage (après compilation/exécution) :

```
$ gcc graph_tab_list_voisins.c
$ ./a.out
[[1],[2,1],[1,0],[1]]
1 voisin de 0  : 1
0 voisin de 1  : 0
deg(1)=2
```