```c
#include "const.h"

/**
 * Algorithme de création de graphe selon les modalités de l'énoncé
 * */

typedef struct {
    int compteur[TAILLE_GRAPHE][3];     // >compte nb voisin vers chaque tier dans la liste
    int proba[TAILLE_GRAPHE][1];        // >nb de noeuds vers le meme tier
} insert;

// Graphe représenté par une liste d'adjacence
typedef struct {
    int list[TAILLE_GRAPHE][TAILLE_GRAPHE];     // >pointe vers 100 listes** qui pointe chacune vers 3 liste* ( qui sont les liste de voisin vers un tier1, tier2, tier3) qui pointe vers une liste
    insert I;                                   // >permet de bien initialiser le graphe
} graphe;

// Génére un graphe
// @return graphe graphe generé
graphe* init_graphe();


// Verifie si le sommet i est dans la liste de j et inversement
//( meme si ca sert a rien car quand on ajoute un voisin a un sommet on
// ajoute aussi le sommet au voisin ) est dans la liste
// @param G le graphe
// @param i sommet
// @param etat_i dans quel tier est le sommet i
// @param j sommet
// @param etat_j dans quel tier est le sommet j
// @return booléens
int verifSiSommetInListe(graphe* G, int i, int j);

// Test si le nb de noeuds max est atteint
// @param G le graphe
// @param i sommet
// @param noeudsMax le noeuds maximum
// @param etat dans quel tier appartient le sommet i
// @return booléens
int test_noeuds_max(graphe* G, int i, int noeudsMax, int etat);

// Calcul quel noeuds choisir
// @param G le graphe
// @param sommet un sommet du graphe
// @param deb debut
// @param fin fin
// @param noeudsMax nb de noeuds max du sommet vers le meme tier
// @param etat_sommet dans quel tier appartient le sommet dans lequelle on cherche un voisin
// @param etat_i dans quel tier appartient le sommet i ( le voisin )
// @return noeuds
int calcul_noeuds(graphe* G, int sommet, int deb, int fin, int noeudsMax, int etat_sommet, int etat_i);
```

```
52
53  // calcul les arc du tier 1
54  // @param G le graphe
55  // @return un graphe géneré
56  void calculTier1(graphe* G);
57
58  // calcul les arc du tier 2
59  // @param G le graphe
60  // @return un graphe géneré
61  void calculTier2(graphe* G);
62
63  // calcul les arc du tier 3
64  // @param G le graphe
65  // @return un graphe géneré
66  void calculTier3(graphe* G);
67
```