INF155 - Enregistrement et allocation dynamique

Exercice 10

Question 1:

Nous souhaitons écrire un programme qui manipule des polygones dans dans le plan (2D). Tout polygone peut être défini par un ensembles de points. Écrire un module t_point, qui définit le type t_point (un point est défini par deux coordonnées x et y réelles) ainsi que les constructeurs, destructeurs, accesseurs et mutateurs suivants:

```
t_point *t_point_init();

t_point *t_point_init_coords(double x, double y);

void t_point_destroy(t_point *p);

double t_point_get_x(const t_point *p);

double t_point_get_y(const t_point *p);

void t_point_set_x(t_point *p, double x);

void t_point_set_y(t_point *p, double y);
```

Question 2:

Complétez le module t_point de la question précédente par les fonctions additionnelles suivantes:

```
double t_point_calculer_distance(const t_point *p1, const t_point *p2);
//Calcule la distance entre les points p1 et p2

void t_point_deplacer(t_point *p, double dx, double dy);
//Déplace le point x d'une distance (dx, dy)
```

Question 3:

En utilisant le type t_point, définissez le module t_vecteur. Un vecteur se définit par deux points (x1,y1) et (x2, y2). Vous devez définir les constructeurs, destructeurs et accesseurs suivants:

```
t_vecteur *t_vecteur_init(double x1, double y1, double x2, double y2);

void t_vecteur_destroy(t_vecteur *vecteur);

t_point *t_vecteur_get_point1(const t_vecteur *vecteur);

t_point *t_vecteur_get_point2(const t_vecteur *vecteur);
```

Question 4:

Complétez votre module t_vecteur en y ajoutant la fonction permettant de faire la translation d'un vecteur d'une distance (dx, dy)

```
void t_vecteur_translater(t_vecteur *vecteur, double dx, double dy);
```

Question 5:

En utilisant le type t_point, définir le type t_polygone ainsi que le constructeur et le destructeur ci-dessous. Un polygone se compose d'un ensemble variable de points. Ainsi le constructeur d'un polygone reçoit en paramètre le nombre de points du polygone à créer.

Note: Vous devez optimiser la mémoire et ne pas en utiliser plus que nécessaire.

```
t_polygone *t_polygone_init(int nb_points);

void t_polygone_destroy(t_polygone *polygone);

int t_polygone_get_nb_points(const t_polygone *polygone);

void t_polygone_set_pointx(t_polygone *polygone, int num_point, double x, double y);
//Définit les coordonnées du point num_point

//Calcule le périmètre du polygone en additionnant la distance entre ses points
double t_polygone_calculer_perimetre(t_polygone *polygone);
```

Question 6:

Complétez votre module t_polygone en ajoutant la fonction permettant d'ajouter un nouveau point à un polygone.

```
void t_polygone_ajouter_point(t_polygone *polygone, double x, double y);
```