TP3 : Structures en C

★ **Exercice 1.**

▷ **Question 1.** Écrire une fonction `symetrique` qui prend en argument :
- l'abscisse ($x \in \mathbb{R}$) d'un point,
- son ordonnée ($y \in \mathbb{R}$),
- et une option parmi XX,YY,ORIG selon laquelle, la fonction *symetrique* modifie les coordonnées d'un point avec celles de son symétrique par rapport à l'axe des x, des y ou par rapport à l'origine respectivement.

On utilise une énumération pour déclarer les différentes options possibles. La fonction retourne 1 si l'opération se déroule correctement, 0 sinon.

▷ **Question 2.** Dans la suite, un point est à définir par une structure *Point* avec les champs suivants :
- id : un nombre entier, identifiant du point,
- x : un nombre réel (float), l'abscisse du point,
- y : un nombre réel (float), l'ordonnée du point.

▷ **Question 3.** Écrire une fonction qui permet d'afficher les différents champs d'une structure *Point* avec le format *id(x,y)*. La fonction prend en argument une structure Point.

▷ **Question 4.** Écrire une fonction qui permet d'initialiser un point et qui prend comme argument une structure Point.

▷ **Question 5.** Récrire la fonction `symetrique` de telle sorte qu'elle prenne en argument en plus de l'option, les coordonnées du point sous forme d'une structure.

▷ **Question 6.** Écrire une fonction qui permet de définir un segment de droite comme un tableau de MAX points situés entre deux points (extrémités) A et B. On utilisera #define pour définir la valeur de MAX.

▷ **Question 7.** Pour afficher l'information sur l'ensemble des points constituant le segment de droite, écrire une fonction `afficheSegment` et qui fait appel à une fonction que vous avez déjà écrite.

▷ **Question 8.** On s'intéresse maintenant à la taille de la structure Point. Afficher la taille de la structure et vérifier bien que sa taille correspond à la somme des tailles des différents éléments qui la composent.

▷ **Question 9.** On définit une structure Point2 similaire à Point mais qui a son identifiant codé comme étant **short**. Même question pour cette nouvelle structure. Que constate -t- on ?

▷ **Question 10.** Afin de comprendre comment la structure est stockée en mémoire, écrire une macro OFFSET(x,Y) qui donne pour un élément x de la structure Y, son adresse relative par rapport à l'adresse de début de la structure. Afficher l'offset de chaque élément dans les 2 structures. Conclure !

★ **Exercice 2.** On souhaite créer un programme en C gérant un annuaire très simplifié qui associe à un nom de personne un numéro de téléphone.

▷ **Question 1.** Créer une structure Personne pouvant contenir ces informations (nom et téléphone). Le nom peut contenir 32 caractères et le numéro 16 caractères.

▷ **Question 2.** Créer une nouvelle structure qui va représenter le carnet d'adresses. Cette structure Carnet contiendra un tableau de 20 Personne et un compteur indiquant le nombre de personnes dans le tableau.

▷ **Question 3.** Créer ensuite une fonction qui renvoie une structure Personne en prenant en argument un nom et un téléphone.

▷ **Question 4.** Rajouter une fonction qui affiche les informations contenues dans la structure Personne passée en argument.

▷ **Question 5.** Créer une fonction qui ajoute une personne dans un carnet.

▷ **Question 6.** Créer une fonction qui affiche un carnet.

▷ **Question 7.** À partir des étapes précédentes, faire programme gérant un carnet d'adresse. Créer un menu qui propose d'ajouter une nouvelle personne, d'afficher le carnet ou de quitter.

Une extension possible serait d'ajouter une fonction de sauvegarde de l'annuaire dans un fichier, et de faire en sorte que les données sauvegardées puissent être lues automatiquement au démarrage.

★ **Exercice 3.** Écrire un programme C qui permette de lire au clavier une suite de caractères représentant la plaque minéralogique (ancien format !) d'un véhicule (comme `982 BZZ 54`) et qui affiche à l'écran les plaques des 20 véhicules immatriculés à la suite de celui-ci (dans le même département).

Ainsi, si la chaîne lue est ≪`998 BZZ 54`≫, alors le programme affichera
≪`999 BZZ 54`≫, ≪`001 CAA 54`≫, ≪`002 CAA 54`≫, ..., ≪`019 CAA 54`≫.