TP 2:
Structures - Unioins

Exercice 1: Structure point

Dans un plan géométrique, un point p est définit par ses deux coordonnées: x et y, et la distance entre deux points $p_1(x_1,y_1)$ et $p_2(x_2,y_2)$ peut être définit par:

$$d = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

Ecrire un programme qui définit une structure point, et créer une fonction qui calcule la distance entre deux points (donnés en arguments)
Faire un test dans main() par $p_1(1,2)$ et $p_2(3,4)$ et cela doit donner: 2.83 comme distance.

- Pour calculer la racine carrée, on dispose de la fonction: sqrt() de la bibliothèque math.h
- Si cette fonction n'est pas reconnue, on peut ajouter l'option **-lm** en ligne de commande, ou dans CodeBlocks: Settings → Compiler → Linker settings → Other linker options.

Exercice 2: Structure triangle

Maintenant, on veut modéliser un triangle contenant 3 points A, B et C. En se basant sur l'exercice précédent, définir une nouvelle structure triangle et une nouvelle fonction qui calcule son périmètre.
Faire un test dans main() par A(1,2), B(3,4) et C(3,4) ce qui donnera: 11.31 comme périmètre.

Exercice3: Structure Personne

Créer un programme contenant une structure *Personne* avec les champs: nom, prenom, genre: masculin / femminun, status: étudiant/employé; s'il est étudiant, il aura le champ: numéro Apogée, s'il est employé, il aura deux informations sur son emploi: le nom de l'établissement et le numéro de somme.
On crée une fonction qui affichera un message d'accueil à une personne (selon son genre), puis présentera toutes ses informations.

Dans *main*, on va créer des instances de la structure *Personnes* puis tester la fonction.
Utiliser les énumérations pour les champs concernés.
Par exemple, on peut initialiser une première variable structure par les données suivantes: {"alami","ahmed",MASC, ETUDIANT, {15203}};

Exercice2: Gestion d'erreur

Ecrire un programme qui gère les erreurs liées à la lecture d'une liste (tableau) d'entiers: la liste à manipuler possède une taille déjà connue. Créer une structure avec deux champs: un tableau d'entier et un entier.
Une fonction *recuperElement* sera alors crée pour récupérer la valeur d'un élément selon son indice. Pour gérer les cas possibles, on crée une énumération contenant: SUCCES, INDEX_ERROR, MEMORY_ERROR, OTHER_ERROR

- Si cette récupération est faite avec succès, on retournera SUCCESS
- Si l'indice est hors liste, on retournera INDEX_ERROR

le prototype de cette fonction est ainsi:
enum CodeErreur obtenirElement(struct Liste *liste, int indice, int *resultat);
sur *main*, créer une liste et tester la fonction.