**Les tuples et les listes**

1. Définir la liste : L = [17, 38, 10, 25, 72]. Effectuer les actions suivantes :
   1. afficher la sous-liste du 2ème au dernier élément
   2. afficher la sous-liste du début au 3ème élément
   3. tester l’appartenance des éléments 10 et 35
   4. trier et afficher la liste
   5. ajouter l’élément 12 à la liste et afficher la liste à nouveau
   6. renverser et afficher la liste
   7. afficher l’indice de l’élément 17
   8. supprimer l’élément 38 et afficher la liste
   9. supprimer et afficher l’élément de l’indice 3
   10. supprimer le premier et le dernier élément et afficher la liste
   11. ajouter à la liste L la liste L1 = [’a’, ’aa’, ’bc’] et afficher la liste L
2. Ecrire une fonction **max**(T) qui returne le plus grand élément présent dans un tuple T reçu comme paramètre.
3. Ecrire une fonction **min**(T) qui returne le plus petit élément présent dans un tuple T reçu comme paramètre.
4. Ecrire une fonction **sum**(T) qui returne la soome des éléments d’un tuple reçu comme paramètre.
5. Ecrire une fonction **index**(T, x) qui retourne le premier indice de l’élément x dans un tuple T reçu comme paramètre.
6. Ecrire une fonction **count**(T, x) qui retourne le nombre des éléments ayant x comme valeur dans un tuple T reçu comme paramètre.
7. Définir le tuple suivant : T = [32, 5, 12, 8, 3, 75, 2, 15], écrire une fonction **pair\_impair**(T) qui génére et retourne deux nouvelles tuples. Le premier tuple T1 un contiendra les nombres pairs du tuple initial et le deuxième tuple T2 contiendra les nombres impairs.
8. Ecrire un programme qui analyse la liste L = [’Mohamed’, ’Nada’, ’Brahim’, ’Samir’, ’Hind’, ’Ibtissam’] pour générer deux nouvelles listes L1 et L2. L’une contiendra les mots comportant moins de 6 caractères, l’autre les mots comportant 6 caractères ou davantage.
9. Ecrire une fonction **anagramme**(L1, L2) qui prend en entrée deux listes L1 et L2, et qui retourne True si les listes sont anagrammes, et False sinon.

Les anagrammes sont des séquences qui se composent exactement des mêmes éléments :

– L1 = [1,2,3,4,5] et L2 = [5,2,4,1,3]

– L1 = [’t’,’a’,’n’,’g’,’e’,’r’] et L2 = [’a’,’r’,’g’,’e’,’n’,’t’]

1. **Recherche dans un tuple** : Ecrire une fonction **dichotomique**(T, k) qui cherche et retourne la position d’un entier k dans un tuple d’entiers T en utilisant la méthode de **recherche dichotomique**.

**Principe :** Soient **T** un tuple déjà trié de **n** entier et **k** le nombre qu’on recherche. On compare le nombre **k** au nombre qui se trouve au milieu du tuple **T**. Si c’est le même, on a trouvé, sinon on recommence sur la première moitié (ou la seconde ) selon que **k**

est plus petit (ou plus grand) que le nombre du milieu du tableau.

1. **Tri par sélection** : Ecrire une fonction qui tri une liste d’entier L avec la méthode de tri par sélection.

**Principe :** consiste à chercher le plus petit élément pour le placer en 1er, puis de

chercher le plus petit élement dans le reste et de le mettre en second, etc...

# Méthode :

* 1. Chercher le minimum de la liste à partir de la case j
  2. Le permuter avec la case j

1. **Tri à bulles** : Ecrire une fonction qui tri une liste d’entier L avec la méthode de tri à bulles.

**Principe :** On commence par i = 0, on compare l’élément L[0] et l’élément L[1], s’il ne

sont pas dans le bon ordre, on les permute, on passe ensuite à l’élément L[1] et l’élément L[2], puis l’élément L[2] et L[3] et ainsi de suite jusqu’au (n-1)*i*è*me* L[n-1] et n*i*è*me* éléments L[n].

# Méthode :

* 1. à partir du début de la liste, s’assurer que chaque paire de case adjacentes vérifient L[ i ] *≤* L[ i+1 ], sinon effectuer les permutations nécessaire.
  2. Répéter l’étape a jusqu’ce qu’il n’y a plus de changements

1. **Tri par insertion** : Ecrire une fonction qui tri une liste d’entier L avec la méthode de tri par insertion.

**Principe :** L’algorithme du est un algorithme qui insère un élément dans une liste

d’éléments déjà triés (par exemple, par ordre croissant).

C’est le tri du joueur de cartes. On fait comme si les éléments à trier étaient donnés un par un, le premier élément constituant, à lui tout seul, une liste triée de longueur 1. On range ensuite le 2*i*è*me* élément pour que les 2 premiers éléments deviennent triés, puis on range le 3*i*è*me* élément pour que les 3 premiers éléments deviennent triés, et ainsi de suite...

1. **Les matrices** : (liste de listes)
   1. Ecrire une fonction **matrice**() qui remplit et affiche la matrice d’entiers M*L,C* (L : nombre de lignes, C : nombre de colonnes).
   2. Ecrire une fonction **matrice\_somme**() qui calcul et affiche, si c’est possible, la matrice somme S de deux matrices M et N lus au clavier. sinon il affiche un message d’erreur.
   3. Ecrire une fonction **matrice\_produit**() qui calcul et affiche, si c’est possible, la matrice produit P de deux matrices M et N. sinon il affiche un message d’erreur.
   4. Ecrire une fonction **matrice\_transpose**() qui calcul et retourne la matrice trans- posé T de la matrice M.
   5. Ecrire une fonction **matrice\_pascale**() qui calcul et retourne la matrice qui repré- sente le triangle de pascale.