TP2

Boucles imbriquées

Exercice 1 (*Deux éléments les plus proches dans une liste*)

1. Écrire une fonction `plus_proche(liste)` prenant comme argument une liste de nombres réels (de longueur supérieure ou égale à 2) et renvoyant les deux valeurs les plus proches.
2. Combien de fois la fonction `abs` a-t-elle été appelée dans la fonction précédente ?
3. Adapter cette fonction pour l'appliquer à une liste de points (donnés par leur coordonnées) avec renvoi des deux points les plus proches (en considérant la distance euclidienne).

Exercice 2 (*Existence de doublons ?*)

On suppose donné une liste `liste` d'éléments quelconques et on souhaite savoir si cette liste contient des doublons. Autrement dit, on souhaite savoir s'il existe deux indices distincts `i` et `j` tels que `liste[i]` et `liste[j]` sont égaux.
Écrire une fonction `doublon(liste)` prenant comme argument une liste et renvoyant le booléen `True` si cette liste contient des doublons et `False` sinon.

Exercice 3 (*Recherche d'un motif dans un texte*)

1. Écrire une fonction `OccurrenceLettre(c,t)` qui affiche tous les indices d'apparition du caractère `c` dans la chaîne de caractère `t`
2. On peut généraliser ce dernier algorithme pour l'appliquer à la recherche d'un motif dans un texte. Pour cela, l'algorithme naïf est le plus simple. Il procède de la manière suivante :
   - Pour chaque position possible du motif dans le texte, on teste si cette position est une occurrence du motif. Ce test est effectué en comparant les caractères les uns après les autres et de gauche à droite. Dès que deux caractères sont différents, la recherche se poursuit en passant à la position suivante.
   - Si tous les caractères du motif sont égaux aux caractères du texte aux positions correspondantes, une occurrence a été trouvée et la position de cette occurrence est retournée.
   - Sinon, la recherche se poursuit en passant à la position suivante.

   Ecrire une fonction `RechercheNaive(m,t)` en implémentant l'algorithme ci-dessus.
3. Dans le pire des cas, combien de comparaisons de caractères sont effectuées ?

Exercice 4 (*Le tri à bulle*)

L'algorithme de tri à bulle consiste à parcourir une liste plusieurs fois jusqu'à ce qu'elle soit triée, en comparant à chaque parcours les éléments consécutifs et en procédant à leur échange s'ils sont mal triés. les étapes sont :

- on parcourt la liste et si deux éléments consécutifs sont rangés dans le désordre, on les échange ;
- si à la fin du parcours au moins un échange à eu lieu, on recommence l'opération ;
- sinon, la liste est triée, on arrête.

Prenons un exemple avec la liste `[13, 19, 14, 8, 5]` :

- Parcours 1 : [13, **14**, **19**, 8, 5] puis [13, 14, **8**, **19**, 5] puis [13, 14, 8, **5**, **19**]
- Parcours 2 : [13, **8**, **14**, 5, 19] puis [13, 8, **5**, **14**, 19]
- Parcours 3 : [**8**, **13**, 5, 14, 19] puis [8, **5**, **13**, 14, 19]
- Parcours 4 : [**5**, **8**, 13, 14, 19]
- Parcours 5 : [5, 8, 13, 14, 19]

1. Appliquer à la main cet algorithme à la liste `[12, 5, 13, 8, 11, 6]`
2. Ecrire une fonction `tri_bulle(liste)` appliquant cet algorithme à la liste donnée en argument. Cette fonction ne renverra rien. On dit que la liste passée en argument est triée en place.
3. Compter le nombre de comparaisons effectuées dans le pire des cas.