Série 4 : Chaines de caractères, pointeurs et allocation dynamique de la mémoire

Exercice 1 : Vérifier si un mot donné (taille max=30) est un palindrome ou non.

Exercice 2 : Une chaîne de caractères w est un *carré* s'il existe une chaîne u telle que $w = uu$ (par exemple ``chercher'' et ``bonbon'' sont des carrés). Afficher si un mot donné est un carré.

Exercice 3:

Soient ***M1*** et ***M2*** deux mots ayant chacun au plus 20 caractères.
a) Donner la déclaration de ***M1*** et ***M2***.

b) En utilisant les fonctions de la librairie **string.h**, écrire une fonction **DIFFER** qui compte le nombre de caractères qui n'apparaissent pas dans ***M2***.

Prototype : int DIFFER (char * , char *)

c) On se donne un mot ***M*** (lu en entrée) et une suite de mots lus successivement. Ecrire le programme qui compte le nombre de mots, dans cette suite, qui ont au plus 3 caractères qui n'apparaissent dans ***M***.

d) Réécrire ce programme et le prototype de **DIFFER** si on suppose que les longueurs de ***M1*** et ***M2*** ne sont pas connus à l'avance.

Exercice 4 :

Écrire une fonction qui retourne l'adresse d'un caractère ***c*** dans une chaîne ***S*** (si ***c*** apparaît dans ***S***) ou ***NULL*** s'il n'apparaît pas.

Prototype: **char * STRCHR(char * , char)**

Exercice 5 :

Soit ***S*** une chaine et ***c*** un caractère donnés. Écrire, sans utiliser les fonctions de la librairie **string.h**, une fonction qui retourne l'adresse de la dernière occurrence de ***c*** dans ***S***.
Exemple : si ***S*****=** "ALKHAW**A**RIZMI" et ***c*****=**'A', la fonction retourne l'adresse du **3ème 'A'** (en gras dans l'exemple).

Prototype: **char * STRRCHR (char *, char)**

Exercice 6 :

En utilisant les fonctions de la librairie **string.h** :

1. Ecrire une fonction qui insère un mot ***T*** dans un mot ***S*** à la position ***i***.
   Prototype: **void STRINSERT(char * *S*, char * *T* , int *i*)**

2. Ecrire une fonction qui vérifie si deux mots **S*1*** et ***S2*** sont anagrammes ou non.

3. Ecrire une fonction qui, étant donnés un mot ***M*** et un sous-mot ***S***, calcule la fréquence d'apparition de ***S*** dans ***M***.
   Prototype: **int FREQSM (char * *M*, char * *S*)**

Exercice 7 :

1. Soient deux chaînes de caractères **S1** et **S2**. Ecrire une fonction **SOUS_CHAINE** qui détermine si S2 est une sous-chaîne de S1.
   Exemple : S1= "abbaaaba<u>aba</u>bbabbb" et S2 = "aba"

2. On considère deux chaînes de caractères **S1** et **S2**. Ecrire une fonction **LONG_PREFIXE** qui détermine le plus long préfixe de S2 contenu dans S1, en utilisant la fonction précédente.
   Exemple : S1 = "abbaaab<u>abbabb</u>b" et S2 = "<u>abbabb</u>aaa"
   Le résultat est « abbabb »

3. Soit une matrice **a[n,m]** (n<=50, m<=80) de caractères, chaque ligne de la matrice contient une chaîne de caractères, et soit une chaîne **S** de longueur <=80. Ecrire un programme qui affiche pour chaque chaîne de la matrice le *plus long préfixe* contenu dans S.

NB : strcpy (ch1, ch2) : copie la chaîne ch2 dans ch1.

strcmp(ch1, ch2) : comparaison lexicographique de deux chaines
retourne 0 si ch1 est identique à ch2
retourne -1 (résultat négatif) si ch1 précède ch2
retourne 1 (résultat positif) si ch1 suit ch2