TD7-alpha

1 - Listes chaînées

Soient les structures :

```
typedef struct _cell{
     int valeur;
     struct _cell *suivant;
} Cellule;
```

```
typedef struct {
     Cellule *tete;
} Liste;
```

Pour les fonctions vous adopterez un style impératif.

a) Ecrire une fonction prenant deux paramètres entiers a et b et qui crée une liste dont les éléments sont les entiers de a à b inclus.
b) Ecrire une fonction prenant en paramètre une liste chaînée l et qui crée une copie (profonde) de cette liste et qui renvoie une référence sur cette liste.
c) Ecrire une fonction prenant en paramètre deux listes chaînées, l1 et l2, et qui "ajoute" l2 à la fin de l1.
d) Ecrire une fonction prenant en paramètre deux listes chaînées, l1 et l2, et qui crée une nouvelle liste chaînée (copie profonde) qui est la concaténation de l1 et l2.
e) Ecrire une fonction qui inverse une liste passée en paramètre (deux versions : sur place et avec copie profonde)
f) Ecrire une fonction prenant en paramètre une liste l1 et un pointeur sur fonction (int -> int) et qui applique la fonction à chaque élément de la liste. Donner un exemple avec une liste dont tous les éléments sont doublés puis mis au carré.

Pointeur sur fonction:

Un pointeur sur fonction se définit comme suit :

```
type_retour (*fct)(type_param1, type_param2,...)
```

Ainsi, pour définir un pointeur sur fonction de type int -> int :

```
int (*fct)(int)
```

L'appel se fait en utilisant le nom de la fonction.

Il est aussi possible de définir des tableaux de pointeurs sur fonction :

```
int (*fct[2])(int)={f1,f2};
```

2 - Tableau de chaîne de caractères

a) Ecrire une fonction prenant en paramètre une chaîne de caractères s, un caractère c et qui renvoie un tableau de chaînes de caractères correspondant au découpage de s suivant le caractère c
b) Ecrire une fonction prenant en paramètre un tableau de chaînes de caractères ts, un caractère c et qui renvoie une chaîne qui est la concaténation des chaînes du tableau séparées par le caractère c

3 - Matrices

Soit le type :

```
typedef struct{
     float**;
     int L;
     int C;
}Matrice;
```

a) Ecrire une fonction qui prend deux entiers l et c et qui crée une Matrice lxc, initialisée à 0.
b) Ecrire une fonction qui prend deux Matrices M et N et qui renvoie une Matrice égale au produit de M et N.