LES LISTES

On a vu en cours la structure de données suivante pour réaliser les listes chaînées sans entête.

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 10

typedef struct noeud *ptrNoeud ;
typedef struct noeud {
                        int valeur ;
                        ptrNoeud suivant;
                    }
                    Noeud ;

typedef ptrNoeud Liste ;

void afficher (Liste l) {
   int i=0;
   printf("\n");
   if (l){
    do {
      printf("Elém %d : %d\n",++i,l->valeur);
      l=l->suivant;
     }
    while (l);
  }
  else printf("La liste est vide");
}

int estVide(Liste l) {
        return (l==NULL);
}
int valeur(Liste l) {
        return (l->valeur);
}

void supprimerPrem(Liste *l) {
   ptrNoeud aSupr=*l;
   *l=(*l)->suivant;
   printf("Je détruis : %d \n",aSup->valeur);
   free(aSup);
```

```
Liste listeCreer() {
        return NULL;
}

 void ajouterFin (int v, Liste *l) {
  /* création d'un nouveau noeud */
  /* ajout en fin de liste */
    ptrNoeud nouv, courant;
    if (nouv=(ptrNoeud) malloc (sizeof(Noeud))) {
      nouv->valeur=v;
      nouv->suivant=NULL;
    /* Premier cas L est vide */
      if (!*l)
        *l=nouv;
      else {
        courant=*l;
        while (courant->suivant)
          courant=courant->suivant;
         courant->suivant=nouv;
      }
    }
    else {
      printf("Erreur d'allocation");
      exit(1);
    }
  }

 void main () {
        Liste l1 ;
        int i;
        l1=listeCreer();
        for (i=0; i<MAX; i++)
                ajouterFin(i,&l1);

        afficher(l1);

        while (!estVide(l1)) {
                int x=valeur (l1);
                printf("%d, ",x);
                supprimerPrem(&l1);
        }

        afficher(l1);
        printf("\n");
}
```

Ecrire les opérations suivantes sur les listes. (utilisation : cf. fichier main)

a) une fonction d'ajout utilisable en notation fonctionnelle : **LajouterEnTete**
b) une fonction de calcul de longueur itérative et récursive : **Llongueur** et **LlongueurR**
c) une fonction de recherche itérative et récursive : **Lrecherche** et **LrechercheR**
d) une fonction de concaténation sans modification : **append**
e) une fonction de concaténation avec modification : **nconc**
f) une fonction de suppression d'un élément itérative et récursive : **Lsupprime** et **LsupprimeR**

Programme de test des différentes fonctions sur les listes

```
void main () {
        Liste l1 ,l2, l3, l4, l5
        Int i;
/* Création des Listes */
        l1=listeCreer();      l2=listeCreer();      l3=listeCreer();      l4=listeCreer();      l5=listeCreer();

/* Initialisation des Listes L1 et L2 */
        for (i=0; i<MAX; i++)
                ajouterFin(i,&l1);
        for (i=0; i<MAX; i++)
                ajouterFin(i*i,&l2);

/* Affichage */
        afficher(l1);
        afficher(l2);

/* Test d'append */
        printf("\nAprès append\n");
                l3=append(l1,l2);
                afficher (l3);
                printf("\nL1:\n");      afficher (l1);      printf("\nL2:\n");      afficher(l2);

/* Test de Nconc */
        printf("\nAprès nconc\n");
                l4=nconc(l1,l2);
                afficher(l4);
                printf("\nL1:\n");      afficher(l1);      printf("\nL2:\n");      afficher(l2);

/* Test de l'ajout fonctionnel */
        for (i=0; i< MAX;i++)
                l5=Lajouter(i*i*i,l5);
        afficher (l5);

/* Test de la recherche */
        if (LrechercheR(27,l5))              printf("J'ai trouvé 27 dans L5");
        if (Lrecherche (64,l5))              printf("\nJ'ai trouvé 64 dans L5");

/* Test Suppression et Longueur */
        printf("\nLongueur de L3 = %d", Llongueur(l3));
        l5=Lsupprimer(27,l5);                        /* au milieu */
        l5=LsupprimerR(729,l5);              /* au début */
        l5=Lsupprimer(0,l5);                         /* a la fin */
        afficher(l5);
        printf("\nLongueurR de L5 = %d", LlongueurR(l5));

        printf("\n"); }
```