Correction TP 6 - Les listes chaînées

Exercice 1

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

typedef struct element * Pelement;
typedef struct liste * Pliste;
typedef struct element{
   int x;
   Pelement suivant;
}Element;

typedef struct liste{
  Pelement premier;
   Pelement courant;
   Pelement dernier;
}Liste;

void initListe(Pliste L){
   L = (Pliste) malloc ( sizeof(Liste));
   L->premier = (Pelement) malloc ( sizeof(Element));
   L->courant = (Pelement) malloc ( sizeof(Element));
   L->dernier = (Pelement) malloc ( sizeof(Element));
   L->premier = NULL;
   L->courant = NULL;
   L->dernier = NULL;
}

void insereUnElemt(Pliste L, Pelement nouveau){
   nouveau->suivant = L->premier;
        L->premier = nouveau;
   if( L->dernier ==NULL )
      L->dernier = nouveau;
}

void creeListeDecroissante(Pliste L, int n){
   printf("***** creeListeDecroissante() *****\n");
   int i;
   Pelement Pel;
   for(i=1; i<=n; i++){
      Pel = (Pelement)malloc( sizeof (Element));
      Pel->x = i;
      insereUnElemt(L, Pel);
   }
}

void afficheListe(Pliste L){
   L->courant = L->premier;
   while (L->courant != NULL){
      printf("%d, ", L->courant->x);
      L->courant = L->courant->suivant;
   }
}

Liste l;maListe = &l;

int main(){
   initListe(maListe);
   creeListeDecroissante(maListe, 5);
   printf("** afficheListe() ** \n[ ");
   afficheListe(maListe);

   printf(" ]\n");
   return 0;
}
```

Exercice 2

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

typedef struct element * Pelement;
typedef struct liste * Pliste;

typedef struct element{
   int x;
   Pelement suivant;
}Element;

typedef struct liste{
  Pelement premier;
   Pelement courant;
   Pelement dernier;
}Liste;

Liste l;
Pliste maListe = &l;

void initListe(Pliste L){
   L = (Pliste) malloc ( sizeof(Liste));
   L->premier = (Pelement) malloc ( sizeof(Element));
   L->courant = (Pelement) malloc ( sizeof(Element));
   L->dernier = (Pelement) malloc ( sizeof(Element));
   L->premier = NULL;
   L->courant = NULL;
   L->dernier = NULL;
}

void insereUnElemt(Pliste L, Pelement nouveau){
   nouveau->suivant = L->premier;
        L->premier = nouveau;
   if( L->dernier ==NULL )
      L->dernier = nouveau;
}

void creeListeDecroissante(Pliste L, int n){
   printf("***** creeListeDecroissante() *****\n");
   int i;
   Pelement Pel;
   for(i=1; i<=n; i++){
      Pel = (Pelement)malloc( sizeof (Element));
      Pel->x = i;
      insereUnElemt(L, Pel);
   }
}

void afficheListe(Pliste L){
   L->courant = L->premier;
   while (L->courant != NULL){
      printf("%d, ", L->courant->x);
      L->courant = L->courant->suivant;
   }
}

float moyenne(Pliste L){
   int som = 0, cpt = 0;
   L->courant = L->premier;
        while (L->courant != NULL){
                cpt++;
      som += L->courant->x;
                L->courant = L->courant->suivant;
        }

   return (som/cpt);
```

```
}

int main(){
   initListe(maListe);
   creeListeDecroissante(maListe, 5);
   printf("Liste  = [ ");
   afficheListe(maListe);
   printf(" ]\n");

   printf("Moyenne(Liste) = %.2f\n", moyenne(maListe));
   return 0;
}
```

Exercice 3

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

typedef struct element * Pelement;
typedef struct liste * Pliste;

typedef struct element{
   int x;
   Pelement suivant;
}Element;

typedef struct liste{
   Pelement premier;
  Pelement courant;
   Pelement dernier;
}Liste;

Liste l, ll;

Pliste maListe = &l;
Pliste pl= &ll;

void initListe(Pliste L){
   L = (Pliste) malloc ( sizeof(Liste));
   L->premier = (Pelement) malloc ( sizeof(Element));
   L->courant = (Pelement) malloc ( sizeof(Element));
   L->dernier = (Pelement) malloc ( sizeof(Element));
   L->premier = NULL;
   L->courant = NULL;
   L->dernier = NULL;
}

void insereUnElemt(Pliste L, Pelement nouveau){
   nouveau->suivant = L->premier;
        L->premier = nouveau;
   if( L->dernier ==NULL )
      L->dernier = nouveau;
}

void creeListeDecroissante(Pliste L, int n){
   printf("***** creeListeDecroissante() *****\n");
   int i;
   Pelement Pel;
   for(i=1; i<=n; i++){
      Pel = (Pelement)malloc( sizeof (Element));
      Pel->x = i;
      insereUnElemt(L, Pel);
   }
}

void afficheListe(Pliste L){
   printf("** afficheListe() **\n");
   L->courant = L->premier;
```

```
   printf("Liste  = [ ");
   while (L->courant != NULL){
      printf("%d, ", L->courant->x);
      L->courant = L->courant->suivant;
   }
   printf(" ]\n");
}

void inserFinListe(Pliste L, Pelement nouveau){
   if (L->dernier ==NULL){
      insereUnElemt(L, nouveau);
   }else{
      nouveau->suivant = L->dernier->suivant;
      L->dernier->suivant = nouveau;
      L->dernier = nouveau;
   }
}

void carres(Pliste L, Pliste Lc){
   Pelement el;
  initListe(Lc);
   L->courant = L->premier;
        while (L->courant != NULL){
      el = (Pelement) malloc(sizeof(Element));
      el->x = pow(L->courant->x, 2);
      inserFinListe(Lc, el);
               L->courant = L->courant->suivant;
        }
}

int main(){
   initListe(maListe);
   creeListeDecroissante(maListe, 5);
   afficheListe(maListe);
   carres(maListe, pl);
   afficheListe(pl);
   return 0;
}
```

Exercice 5

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

typedef struct element * Pelement;
typedef struct liste * Pliste;

typedef struct element{
   int x;
   Pelement suivant;
}Element;

typedef struct liste{
  Pelement premier;
   Pelement courant;
   Pelement dernier;
}Liste;

void initListe(Pliste L){
   L = (Pliste) malloc ( sizeof(Liste));
   L->premier = (Pelement) malloc ( sizeof(Element));
   L->courant = (Pelement) malloc ( sizeof(Element));
   L->dernier = (Pelement) malloc ( sizeof(Element));
   L->premier = NULL;
   L->courant = NULL;
   L->dernier = NULL;
}
```

```
void insereUnElemt(Pliste L, Pelement nouveau){
   nouveau->suivant = L->premier;
        L->premier = nouveau;
   if( L->dernier ==NULL )
      L->dernier = nouveau;
}

void creeListeDecroissante(Pliste L, int n){
   printf("***** creeListeDecroissante() *****\n");
   int i;
   Pelement Pel;
   for(i=1; i<=n; i++){
      Pel = (Pelement)malloc( sizeof (Element));
      Pel->x = i;
      insereUnElemt(L, Pel);
   }
}

void afficheListe(Pliste L){
   L->courant = L->premier;
   printf("Liste = [ ");
   while (L->courant != NULL){
      printf("%d, ", L->courant->x);
      L->courant = L->courant->suivant;
   }
   printf(" ]\n");
}

void supprimePremier(Pliste L){
   Pelement el = L->premier;
   L->premier = L->premier->suivant;
   free(el);
   el=NULL;
}
```

```
}

Liste l;
Pliste maListe = &l;

int main(){
   initListe(maListe);
   creeListeDecroissante(maListe, 5);
   afficheListe(maListe);
   supprimePremier(maListe);
   afficheListe(maListe);
   return 0;
}
```

Exercice 6

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

typedef struct element * Pelement;
typedef struct liste * Pliste;

typedef struct element{
   int x;
   Pelement suivant;
}Element;

typedef struct liste{
  Pelement premier;
   Pelement courant;
   Pelement dernier;
}Liste;

void initListe(Pliste L){
```

```
   L = (Pliste) malloc ( sizeof(Liste));
   L->premier = (Pelement) malloc ( sizeof(Element));
   L->courant = (Pelement) malloc ( sizeof(Element));
   L->dernier = (Pelement) malloc ( sizeof(Element));
   L->premier = NULL;
   L->courant = NULL;
   L->dernier = NULL;
}

void insereUnElemt(Pliste L, Pelement nouveau){
   nouveau->suivant = L->premier;
        L->premier = nouveau;
   if( L->dernier ==NULL )
      L->dernier = nouveau;
}

void creeListeDecroissante(Pliste L, int n){
   printf("***** creeListeDecroissante() *****\n");
   int i;
   Pelement Pel;
   for(i=1; i<=n; i++){
      Pel = (Pelement)malloc( sizeof (Element));
      Pel->x = i;
      insereUnElemt(L, Pel);
   }
}

void afficheListe(Pliste L){
   L->courant = L->premier;
   printf("Liste = [ ");
   while (L->courant != NULL){
      printf("%d, ", L->courant->x);
      L->courant = L->courant->suivant;
   }
   printf(" ]\n");
}

void supprimeDernier(Pliste L){
   Pelement el = L->dernier;
   Pelement avDernier;
   L->courant = L->premier;
   while(L->courant->suivant->suivant !=NULL){
      L->courant = L->courant->suivant;
  }
   avDernier = L->courant;
   free(avDernier->suivant);
   avDernier->suivant = NULL;
   L->dernier = avDernier;
}

Liste l;
Pliste maListe = &l;

int main(){
   initListe(maListe);
   creeListeDecroissante(maListe, 5);
   afficheListe(maListe);
   supprimeDernier(maListe);
   afficheListe(maListe);
   return 0;
}
```

Exercice 8

```
#include <stdio.h>
#include <stdlib.h>
/* Gestion d'une pile FIFO */
#define MAX 5
```

```c
typedef struct element * Pelement;
typedef struct liste * Pliste;
struct element{
   int n;
   Pelement suivant;
}Element;

struct liste{
   Pelement premier;
   Pelement courant;
   int taille;
}Liste;

struct liste l;
Pliste pile = &l;

void initialiserPile(Pliste P){
   P = (Pliste)malloc(sizeof(Liste));
   P->premier =
(Pelement)malloc(sizeof(Element));
   P->courant =
(Pelement)malloc(sizeof(Element));
   P->premier = NULL;
   P->courant = NULL;
   P->taille = 0;
}

int pileVide(Pliste P){
   return (P->premier == NULL);
}

void empiler(Pliste P, int i){
   Pelement
nouveau=(Pelement)malloc(sizeof(Element));
   nouveau->n = i;
   if ( pileVide(P) ){
      nouveau->suivant = NULL;
      P->premier = nouveau;
   } else {
      nouveau->suivant = P->premier;
      P->premier = nouveau;
   }
   P->taille++;
}

void depiler(Pliste P){
   if ( !pileVide(P) ){
      Pelement asupprimer;
      asupprimer = P->premier;
      P->premier = P->premier->suivant;
      printf("Element retire de la pile : %d\n",
P->premier->n);
      free(asupprimer);
      asupprimer = NULL;
   }
}

void afficher(Pliste P){
   P->courant=P->premier;
   printf("Pile = ( ");
   while (P->courant != NULL){
      printf("%d, ",P->courant->n);
      P->courant = P->courant->suivant;
   }
   printf(")\n");
}
```

```
void creerPile(Pliste P){
   int i, tab[MAX]= {3,4,5,6,7};
   for (i = 0; i<MAX; i++){
                empiler(pile, tab[i]);
        }
}

int menu(){
   int choix, a;
   printf("*** Gestion d'une pile ***\n");
   printf(" 1 - Initialiser\n");
   printf(" 2 - afficher\n");
   printf(" 3 - Empiler \n");
   printf(" 4 - Depiler\n");
   printf(" 5 - Quitter\n");
   printf("Saisissez votre choix : ");
   scanf("%d",&choix);
   switch (choix){
   case 1:
      initialiserPile(pile);
      printf("Pile initialisee\n");
      menu();
      break;
   case 2:
      afficher(pile);
      menu();
      break;
   case 3:
      printf("Saisir entier a inserer dans la
pile : ");
      scanf("%d", &a);
      empiler(pile, a);
      menu();
      break;
   case 4:
      depiler(pile);
      menu();
      break;
   case 5:
      exit(0);
   default:
      printf("/!\\ Choix non valide /!\\ \n");
      menu();
   }

}

int main(){
/* initialiserPile(pile);
   creerPile(pile);
   afficher(pile);
   depiler(pile);
   afficher(pile);
*/
   menu();
   return 0;
}
```