Correction Atelier 2 : Les listes simplement chainée en C

Compétences à atteindre

C1. Savoir déclarer une liste simplement chainée

C2. Développer des programmes qui insèrent (en tête, à la fin, à une position donnée) des éléments dans la liste

C3. Développer des programmes qui recherchent un élément de la liste, selon sa valeur ou son rang

C4. Développer des programmes qui suppriment un élément de la liste

Soit à manipuler une liste simplement chaînée. Cette liste est composée par un ensemble de cellules contenant les informations des étudiants qui sont chaînées par un pointeur (suivant). Chaque étudiant est caractérisé par son numéro d'inscription (entier) son nom (chaine de caractères), son prénom (chaine de caractères) et sa moyenne (réel).

1. Définir la structure **Etudiant** et la structure **Liste.**
2. Ecrire une fonction **saisieEtudiant et afficherEtudiant** pour la saisie et l'affichage d'un étudiant.
3. Ecrire une fonction d'initialisation de la liste pointée par un pointeur L. L est l'adresse d'une structure de type LISTE **: void init_Liste (Liste * L)**.
4. Ecrire une fonction qui teste et retourne si une liste est vide ou non :

   int listeVide(Liste L).
5. Écrire la fonction Ajout_Debut qui ajoute un étudiant au début de la liste : **void Ajout_Debut(Liste * L, Etudiant etud)**.
6. Écrire la fonction Affiche_Etudiants qui affiche tous les étudiants de la liste **void**

   Affiche_Etudiants (Liste L).
7. Écrire la fonction Ajout_Fin qui ajoute un étudiant à la fin de la liste : **void Ajout_Fin (Liste * L, Etudiant etud)** .
8. Écrire la fonction Recherche_Etudiant qui cherche un étudiant à partir de son numéro d'inscription. Cette fonction retourne l'adresse de cet étudiant s'il est trouvé et NULL si non : **cellule * Recherche_Etudiant(Liste L, int Numins)**.
9. Écrire la fonction Supprime_Etudiant qui supprime un étudiant de la liste sachant son

Num d'inscription : **void Supprime_Etudiant(Liste * L,int Numins)**.

10. Écrire la fonction **Permute_Etudiant** qui permet de permuter deux étudiants.
11. Écrire la fonction Permute_Liste qui permet de permuter la position de 2 étudiants sachant leurs numéros d'inscription**: void permut_Liste(Liste * L, int num1, int num2);.**
12. Écrire un programme principal qui :

- Saisit le nombre d'étudiants de la liste.
- Crée la liste des étudiants en utilisant la fonction Ajout_Debut.
- Affiche tous les étudiants de la liste.
- Ajoute un étudiant en fin de la liste.
- Affiche tous les étudiants de la liste.
- Cherche un étudiant et le supprime de la liste.
- Affiche la nouvelle liste.

Correction

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

typedef struct {
int numIns;
char nom[20];
char prenom[20];
float moyenne;
}etudiant;

typedef  struct cellule {
    etudiant information ;
struct    cellule * suivant ;
} cellule ;

typedef cellule* liste;
etudiant saisieEtudiant();
void afficherEtudiant(etudiant e);
void init_liste(liste*);
int liste_vide(liste);

cellule * recherche_etudiant(liste, int);
int longueur_liste(liste);

void affiche_etudiants(liste);
```

```
void ajout_debut(liste* , etudiant);

void ajout_fin(liste* , etudiant);
void permuter_etudiant(etudiant *,etudiant *);
void permuter_liste(liste *,int,int);

void saisieEtudiant(etudiant *e){

    puts("Donner le num inscrip: ");
    scanf("%d",&e->numIns);

    printf("Donner le nom: ");
    //gets(E1.nom);
    scanf(" %[^\n]s",e->nom);

    //gets(E1.nom);
    printf("Donner le prenom: ");
   // gets(E1.prenom);
    scanf(" %[^\n]s",e->prenom);

    printf("nDonner la moyenne: ");

    scanf("%f",&e->moyenne);

}
void afficherEtudiant(etudiant e){
printf("Etudiant %s %s de num inscri %d a comme moyenne
%.2f.\n",e.nom,e.prenom,e.numIns,e.moyenne);
}
int vide_liste(liste l){
return l == NULL ;
}


cellule* recherche_etudiant(liste l , int numInsc){

cellule * p;

if(vide_liste(l))
return NULL;

p = l;

while(p != NULL){

if(p->information.numIns == numInsc)
return p;

    p=p->suivant;
}

return NULL;
```

```
}

int longueur_liste(liste l){

cellule * p;
int longueur=0;


while(l != NULL){
longueur++;
l = l->suivant;
}
return longueur;
}

void afficherEtudiants(liste l){
while(l!=NULL){

afficherEtudiant(l->information);
printf("Etudiant %s %s de num inscri %sd a comme moyenne
l = l->suivant;

}
}
void afficher_Rec(liste l){

if(l !=NULL){
printf("%d\n" , l->information);
afficher_Rec(l->suivant);
}
}

void ajout_debut(liste *l , etudiant e){

cellule * n;

n = (cellule*)malloc(sizeof(cellule));
n->information = e;
n->suivant = *l;

*l = n;
}

void ajout_fin(liste *l , etudiant e){

cellule * n;


n = (cellule*)malloc(sizeof(cellule));
n->information = e; n->suivant = NULL;
 if(vide_liste(*l))
   *l = n;
else{
```

```
cellule * p = *l;
while(p->suivant != NULL)
   p = p->suivant;
p->suivant = n;
}
}
void supprimer_tete_liste(liste *l){
if(!vide_liste(*l)){
cellule * p = *l;
*l = p->suivant;
free(p);
}
}

void supprimer_element(liste *L,int numIns)
{
cellule *p1, *p2 ;
p1=*L ;
p2=*L ;
while(p1 !=NULL &&p1->information.numIns !=numIns)
{
p2=p1 ;
p1=p1->suivant;
}
if( p1==*L){
    supprimer_tete_liste(L);
    printf("Suppression faite");
    return;
}
if(p1!=NULL)
 {
  p2->suivant=p1->suivant;
  free(p1);
  printf("Suppression faite");
}
else
    printf("non trouve");
}

void permuter_etudiant (etudiant *e1,etudiant *e2){
    etudiant aux;
    aux=*e1;
    *e1=*e2;
    *e2=aux;
}

void permuter_liste(liste *L,int num1,int num2){

    cellule *p1=recherche_etudiant(*L,num1);
    cellule *p2=recherche_etudiant(*L,num2);
if(p1&&p2)
    {
        permuter_etudiant(&p1->information,&p2->information);
```

```
    }
}


int main(){
liste L;
int nb,num;
etudiant e;
init_liste(&L);
printf("Donner la taille de la liste: ");
scanf("%d",&nb);
printf("Ajout des elements de la liste\n");
for(int i=0;i<nb;i++){
    e=saisieEtudiant();
    ajout_debut(&L,e);
}

printf("Affichage des elements de la liste\n");
affiche_etudiants(L);
printf("Donner le num d'incription de l'etudiant à chercher");
scanf("%d",&num);
cellule *p=recherche_etudiant(L,num);

if(p==NULL)
    printf("Etudiant non trouve");
else
{
   printf(("Etudiant trouve. Ses informations sont :") ;
   afficherEtudiant(p->info) ;
   supprimer_element(&L,num);

}

Affiche_etudiants(L);
return 0 ;
}
```