TP Liste Linéaire Chainée ou «linked lists»

Un polynôme peut être représenté par une LLC.

1. Définir **la structure** des éléments ainsi que **le modèle LLC**.
2. Ajouter une méthode qui calcule la dérivée de ce polynôme
3. Ajouter une méthode qui calcule la somme de deux polynômes
4. Ajouter une méthode qui calcule le produit de deux polynômes
5. Calcule la valeur de polynôme dans un point donnée
6. Ecrire un programme qui supprime un élément

Déclaration de la structure de données :

```
struct terme
{
        int coeff;
        int pow;
        struct terme *Suiv;
} ;

struct polynome
{
        struct terme *tete;
        struct terme *queu;
        int taille;
};

/*-------------------------------------------------------------*/
/*         Procédures d'implémentation du modèle           */
/*-------------------------------------------------------------*/


struct terme *Allouer ( )
{
return ((terme *) malloc( sizeof(struct terme)));
}

void Affadr(terme *P, terme *Q)
{ P->Suiv = Q;}
void Affterme (terme *P, int coeff, int  pow)
{ P-> coeff = coeff; P-> pow = pow;  }


struct terme *Suivant(terme *P)
{ return( P->Suiv ); }
```

Création de la structure de données :

```
struct terme *Allouer ( )
```

```
{
return ((terme *) malloc( sizeof(terme)));
}
```

Créer polynôme :

```
void cree_polynome (polynome *P)
{
        terme *V, *Q;
        int i;
        printf("Veuillez introduire le nombre de termes du polynome :");
        scanf("%d",&P->taille);
        P->tete =NULL;
        for (i=0; i < P->taille; i++)
        {       V= Allouer ();
                aff_terme(V);
                if (P->tete != NULL) Affadr(Q,V);
                else (P->tete = V);
                Q=V;
        }
        Affadr(Q,NULL);
        P->queu = Q;
}
```

Méthode Afficher polynôme

```
void affich_poly (struct polynome L)
{
}
```

Méthode valeur de polynôme dans un point donnée

```
float calcul_un_point(struct polynome L)
{
}
```

Méthode dérivée d'un polynôme

```
void derivee ( struct polynome *P, struct polynome *S)
{
}
```

Méthode somme de deux polynômes

```
void Somme(struct polynome *P1, struct polynome *P2, struct polynome *P_somme)
{
}
```

Méthode produit de deux polynômes

```
void Produit (struct polynome *P1, struct polynome *P2, struct polynome *P_produit)
{
}
```

Méthode supprimer un élément de polynômes

```
void SuppElt ( struct polynome *P, struct terme *S)
{
}
```

C Program to Implement Binary Tree using Linked List

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

struct node {
   struct node * left;
   char data;
   struct node * right;
};

struct node *constructTree( int );
void inorder(struct node *);

char array[ ] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', '\0', '\0', 'H' };
int leftcount[ ] = {  1,   3,   5,   -1,   9,  -1,  -1,   -1,   -1,  -1 };
int rightcount[ ] = {  2,   4,   6,   -1,  -1,  -1,  -1,   -1,   -1,  -1 };

void main() {
   struct node *root;
   root = constructTree( 0 );
   printf("In-order Traversal: \n");
   inorder(root);
}

struct node * constructTree( int index ) {
   struct node *temp = NULL;
   if (index != -1) {
      temp = (struct node *)malloc( sizeof ( struct node ) );
      temp->left = constructTree( leftcount[index] );
      temp->data = array[index];
      temp->right = constructTree( rightcount[index] );
   }
   return temp;
}

void inorder( struct node *root ) {
   if (root != NULL) {
      inorder(root->left);
      printf("%c\t", root->data);
      inorder(root->right);
   }
}
```