Programmation II : SMI (S4)

TP N° 5 : Les listes chainées (Correction)

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

typedef struct Monome * Polynome;
typedef struct Monome{
       float coeff;
       int   puiss;
       Polynome next;
}Monome;

Monome * creerMonome(float coef, int exp){
    Monome * mono;
    mono = (Monome*)malloc(sizeof(Monome));
    mono->coeff = coef;
    mono->puiss = exp;
    mono->next = NULL;
    return mono;
}

Polynome saisirPolynome(){
    Polynome P = NULL;
    Monome * m;
    float c;
    int p;
    printf("Saisie du polynome : \n");
    do{
        printf("Coeffecient : ");
        scanf("%f", &c);
        printf("Puissance : ");
        scanf("%d", &p);
        if(p>=0){
            m = creerMonome(c, p);
            m->next = P;
            P = m;
        }
    }while(p>=0);
    return P;
}

void afficherMonome(Monome * m){
    printf("+ %.2fX^%d ", m->coeff, m->puiss);
}

void afficherPolynome(Polynome P){
    printf("\n");
    while(P!=NULL){
        afficherMonome(P);
        P = P->next;
    }
    printf("\n");
}

float evaluerPolynome(Polynome P, float x){
    float val = 0;
    while(P!=NULL){
        val += P->coeff * pow(x, P->puiss);
        P = P->next;
    }
    return val;
}

Polynome deriverPolynome(Polynome P){
    Polynome DP = NULL;
    Monome * m;
    float cc;
    int pp;
    while(P!=NULL){
        float cc = P->coeff;
        pp = P->puiss;
        if(pp>0){
            cc = cc * pp; // ou bien  cc *= pp;
            pp = pp - 1;  // ou bien  pp--;
            m = creerMonome(cc, pp);
            m->next = DP;
            DP = m;
        }
        P = P->next;
    }
    return DP;
}

void simplifierPolynome(Polynome X){
    if(X!=NULL && X->next!=NULL){
        Polynome P = X;
        Polynome Q, prec;
        Monome * tmp;
        while(P->next != NULL){
            prec = P;
            Q = P->next;
            while(Q != NULL){
                if(P->puiss == Q->puiss){
                    P->coeff += Q->coeff;
                    tmp = Q;
                    prec->next = Q->next;
                    Q = Q->next;
                    free(tmp);
                }
                else{
                    Q = Q->next;
                    prec = prec->next;
                }
            }
            if(P->next != NULL){ // On controle si le dernier noeud a ete supprime
                P = P->next;
            }
            printf("\n");
        }
    }
}

Polynome sommePolynome(Polynome P, Polynome Q){
    if(P == NULL && Q != NULL){
        return Q;
    }
    if(P != NULL && Q == NULL){
        return P;
    }
    Polynome som = P;
    while(P->next != NULL){
        P = P->next;
    }
    P->next = Q;
    simplifierPolynome(som);
    return som;
}

Polynome produitPolynomeMonome(Polynome P, Monome m){
    Polynome resultat = NULL;
    Monome * mono;
    float c;
    int p;
    while(P != NULL){
        c = P->coeff * m.coeff;
        p = P->puiss + m.puiss;
        mono = creerMonome(c, p);
        mono->next = resultat;
        resultat = mono;
        P = P->next;
    }
    simplifierPolynome(resultat);
    return resultat;
}

Polynome produitPolynomes(Polynome P, Polynome Q){
    Polynome resultat = NULL;
    while(Q != NULL){
        resultat = sommePolynome(resultat, produitPolynomeMonome(P, *Q));
        Q = Q->next;
    }
    simplifierPolynome(resultat);
    return resultat;
}

main(){
    // Test de saisie et de simplification
    Polynome A;
    A = saisirPolynome();
    afficherPolynome(A);
    simplifierPolynome(A);
    afficherPolynome(A);

    // Test d'evaluation
    printf("%.2f\n", evaluerPolynome(A, 3));

    // Test de dérivation
    Polynome B;
    B = deriverPolynome(A);
    afficherPolynome(B);

    // Test de la somme de deux polynomes
    Polynome C;
    C = sommePolynome(A, B);
    afficherPolynome(C);

    // Test produit de polynome x un monome
    Polynome X;
    Monome mo = * creerMonome(1,2);
    X = produitPolynomeMonome(A, mo);
    afficherPolynome(A);
    afficherPolynome(X);

    // Test produit polynome x polynome
    Polynome Z;

    Z = produitPolynomes(A,B);
    afficherPolynome(Z);
    getch();
}
```