**/\*Write a program to store a polynomial using linked list. Also, perform addition and subtraction on two polynomials\*/**

**#include<iostream>**

**#include<cstdlib>**

**#include<cmath>**

**using namespace std;**

**struct node**

**{**

**int check;**

**int info,xp;**

**node \*next;**

**};**

**class POLY**

**{**

**node \*START;**

**public:**

**POLY():START(NULL) {}**

**void AddExpression(int,int);**

**POLY operator + (POLY &);**

**POLY operator - (POLY &);**

**bool DisplayExpression();**

**};**

**void POLY::AddExpression(int num,int x)**

**{**

**node \*temp=new node;**

**if(temp==NULL)**

**cout<<"\n\nFailed to initialize the memory for new block.\n\n";**

**else**

**{**

**temp->info=num;**

**temp->xp=x;**

**temp->next=NULL;**

**if(START==NULL)**

**START=temp;**

**else**

**{**

**node \*ptr;**

**ptr=START;**

**while(ptr->next!=NULL)**

**ptr=ptr->next;**

**ptr->next=temp;**

**}**

**}**

**}**

**POLY POLY::operator+(POLY &second)**

**{**

**POLY t;**

**if(START==NULL)**

**{**

**cout<<"\n\nThere is no first polynomial expression.\n\n";**

**return t;**

**}**

**else if(second.START==NULL)**

**{**

**cout<<"\n\nThere is no second polynomial expression.\n\n";**

**return t;**

**}**

**else**

**{**

**int c;**

**node \*p1,\*p2;**

**p1=START;**

**while(p1!=NULL)**

**{**

**c=0;**

**p2=second.START;**

**while(p2!=NULL)**

**{**

**if(p1->xp==p2->xp)**

**{**

**c=1;**

**p2->check=1;**

**t.AddExpression((p1->info+p2->info),p1->xp);**

**break;**

**}**

**p2=p2->next;**

**}**

**if(c==0)**

**t.AddExpression(p1->info,p1->xp);**

**p1=p1->next;**

**}**

**p2=second.START;**

**while(p2!=NULL)**

**{**

**if(p2->check!=1)**

**t.AddExpression(p2->info,p2->xp);**

**p2=p2->next;**

**}**

**return t;**

**}**

**}**

**POLY POLY::operator-(POLY &second)**

**{**

**POLY t;**

**if(START==NULL)**

**{**

**cout<<"\n\nThere is no first polynomial expression.\n\n";**

**return t;**

**}**

**else if(second.START==NULL)**

**{**

**cout<<"\n\nThere is no second polynomial expression.\n\n";**

**return t;**

**}**

**else**

**{**

**int c;**

**node \*p1,\*p2;**

**p1=START;**

**while(p1!=NULL)**

**{**

**c=0;**

**p2=second.START;**

**while(p2!=NULL)**

**{**

**if(p1->xp==p2->xp)**

**{**

**c=1;**

**p2->check=1;**

**t.AddExpression((p1->info-p2->info),p1->xp);**

**break;**

**}**

**p2=p2->next;**

**}**

**if(c==0)**

**t.AddExpression(p1->info,p1->xp);**

**p1=p1->next;**

**}**

**p2=second.START;**

**while(p2!=NULL)**

**{**

**if(p2->check!=1)**

**t.AddExpression(-p2->info,p2->xp);**

**p2=p2->next;**

**}**

**return t;**

**}**

**}**

**bool POLY::DisplayExpression()**

**{**

**if(START==NULL)**

**{**

**cout<<"\n\nNo expression\n";**

**return false;**

**}**

**else**

**{**

**node \*ptr;**

**ptr=START;**

**cout<<"\n\nThe expression is :\n";**

**while(ptr!=NULL)**

**{**

**if(ptr==START &&ptr->info>=0)**

**cout<<ptr->info<<"x^"<<ptr->xp<<" ";**

**else if(ptr->info>=0)**

**cout<<"+"<<ptr->info<<"x^"<<ptr->xp<<" ";**

**else**

**cout<<ptr->info<<"x^"<<ptr->xp<<" ";**

**ptr=ptr->next;**

**}**

**cout<<"\n\n";**

**return true;**

**}**

**}**

**int main()**

**{**

**POLY e1,e2,e3;**

**int choice,info,x,y,z;**

**char ch;**

**while(1)**

**{**

**cout<<"1. Enter the first expression\n2. Enter the second expression\n3. Add first and second expressions\n4. Subtract second expression from first expression\n5. Display first expression\n6. Display second expression\n7. Exit\n\nEnter your choice : ";**

**cin>>choice;**

**switch(choice)**

**{**

**case 1:**

**{**

**char c='y';**

**while(c=='y'||c=='Y')**

**{**

**cout<<"\nEnter in the form (coeff,x pow): ";**

**cin>>ch>>info>>ch>>x>>ch;**

**e1.AddExpression(info,x);**

**cout<<"\nWant to add another term for first expression? y/n\n";**

**cin>>c;**

**}**

**break;**

**}**

**case 2:**

**{**

**char c='y';**

**while(c=='y'||c=='Y')**

**{**

**cout<<"\nEnter in the form (coeff,x pow): ";**

**cin>>ch>>info>>ch>>x>>ch;**

**e2.AddExpression(info,x);**

**cout<<"\nWant to add another term for second expression? y/n\n";**

**cin>>c;**

**}**

**break;**

**}**

**case 3:**

**{**

**e3=e1+e2;**

**bool b=e3.DisplayExpression();**

**break;**

**}**

**case 4:**

**{**

**e3=e1-e2;**

**bool b=e3.DisplayExpression();**

**break;**

**}**

**case 5:**

**{**

**bool b=e1.DisplayExpression();**

**break;**

**}**

**case 6:**

**{**

**bool b=e2.DisplayExpression();**

**break;**

**}**

**default :**

**exit(0);**

**}**

**}**

**return 0;**

**}**

**/\*Write a program to store a polynomial using linked list. Also, perform addition and subtraction on two polynomials\*/**

**#include<iostream>**

**#include<cmath>**

**using namespace std;**

**struct Term**

**{**

**float coef;**

**float x\_index;**

**float y\_index;**

**float z\_index;**

**Term \*next;**

**};**

**class Polynomial**

**{**

**public:**

**Term \*head;**

**Polynomial()**

**{**

**head = NULL;**

**}**

**void assign\_polynomial(float c, float x, float y, float z)**

**{**

**Term \*newTerm= new Term;**

**newTerm->coef = c;**

**newTerm->x\_index = x;**

**newTerm->y\_index = y;**

**newTerm->z\_index = z;**

**if(head == NULL)**

**{**

**head = newTerm;**

**head->next = NULL;**

**}**

**else**

**{**

**Term \*ptr = head;**

**while(ptr->next != NULL)**

**{**

**ptr = ptr->next;**

**}**

**ptr->next = newTerm;**

**newTerm->next = NULL;**

**}**

**}**

**Polynomial& operator+(Polynomial& P2)**

**{**

**Polynomial P3;**

**Term \*ptr2 = P2.head;**

**while(ptr2 != NULL)**

**{**

**Term \*ptr1 = head;**

**while(ptr1 != NULL)**

**{**

**if(ptr1->x\_index == ptr2->x\_index && ptr1->y\_index == ptr2->y\_index && ptr1->z\_index == ptr2->z\_index )**

**{**

**ptr1->coef = ptr1->coef + ptr2->coef;**

**break;**

**}**

**ptr1 = ptr1->next;**

**}**

**if(ptr1 == NULL)**

**{**

**this->assign\_polynomial((ptr2->coef),ptr2->x\_index, ptr2->y\_index, ptr2->z\_index);**

**}**

**ptr2 = ptr2->next;**

**}**

**return \*this;**

**}**

**Polynomial& operator-(Polynomial& P2)**

**{**

**Polynomial P3;**

**Term \*ptr2 = P2.head;**

**while(ptr2 != NULL)**

**{**

**Term \*ptr1 = head;**

**while(ptr1 != NULL)**

**{**

**if(ptr1->x\_index == ptr2->x\_index && ptr1->y\_index == ptr2->y\_index && ptr1->z\_index == ptr2->z\_index )**

**{**

**ptr1->coef = ptr1->coef - ptr2->coef;**

**break;**

**}**

**ptr1 = ptr1->next;**

**}**

**if(ptr1 == NULL)**

**{**

**this->assign\_polynomial((ptr2->coef),ptr2->x\_index, ptr2->y\_index, ptr2->z\_index);**

**}**

**ptr2 = ptr2->next;**

**}**

**return \*this;**

**}**

**void display()**

**{**

**Term \*ptr = head;**

**while(ptr != NULL)**

**{**

**if(ptr->coef >= 0)**

**{**

**cout<<"+("<<ptr->coef<<"(x^"<<ptr->x\_index<<")(y^"<<ptr->y\_index<<")(z^"<<ptr->z\_index<<")) ";**

**}**

**else**

**{**

**cout<<"("<<ptr->coef<<"(x^"<<ptr->x\_index<<")(y^"<<ptr->y\_index<<")(z^"<<ptr->z\_index<<")) ";**

**}**

**ptr = ptr->next;**

**}**

**}**

**};**

**int main()**

**{**

**Polynomial POLY1,POLY2;**

**char choice ='y';**

**char trash;**

**float coef,x\_index,y\_index,z\_index;**

**cout<<endl<<endl<<"First polynomial"<<endl;**

**while(true)**

**{**

**cout<<"Enter coef, x\_index, y\_index, z\_index: "<<endl;**

**cin>>coef>>trash>>x\_index>>trash>>y\_index>>trash>>z\_index;**

**POLY1.assign\_polynomial(coef,x\_index,y\_index,z\_index);**

**cout<<endl<<"Add more?(y/n): ";**

**cin>>choice;**

**if(choice == 'n')**

**{**

**break;**

**}**

**}**

**choice = 'y';**

**cout<<endl<<endl<<"Second polynomial: "<<endl;**

**while(true)**

**{**

**cout<<endl<<"Enter coef, x\_index, y\_index, z\_index: "<<endl;**

**cin>>coef>>trash>>x\_index>>trash>>y\_index>>trash>>z\_index;**

**POLY2.assign\_polynomial(coef,x\_index,y\_index,z\_index);**

**cout<<endl<<"Add more?(y/n): ";**

**cin>>choice;**

**if(choice == 'n')**

**{**

**break;**

**}**

**}**

**cout<<"1. Add"<<endl;**

**cout<<"2. Subtract"<<endl;**

**cout<<"3. Exit"<<endl;**

**int option;**

**cout<<"Enter the option:";**

**cin>>option;**

**switch(option)**

**{**

**case 1:**

**POLY1 = POLY1 + POLY2;**

**cout<<endl<<"Sum= ";**

**POLY1.display();**

**break;**

**case 2:**

**POLY1 = POLY1 - POLY2;**

**cout<<endl<<"Difference= ";**

**POLY1.display();**

**break;**

**case 3:**

**exit(1);**

**break;**

**default:**

**cout<<endl<<"Error input"<<endl;**

**break;**

**}**

**return 0;**

**}**