**ROLL NO:-45**

**NAME : Harshit Atul Chilvirwar**

**PRACTICAL NO:-**

**PRACTICAL NAME :- IMPLEMENTATION OF POLYNOMIAL ADDITION /SUBTRACTION**

**(LINKED LIST).**

#include "iostream.h"

#include "conio.h"

class NODE

{

public:

int coeff;

int power;

NODE \*link;

};

class POLYEXPR

{

NODE \*PE1,\*PE2,\*PE3,\*PE4;

int order;

public:

POLYEXPR(int);

void READ\_POLYEXPR1();

void READ\_POLYEXPR2();

void ADD\_POLYEXPR();

void SUB\_POLYEXPR();

void VIEW\_POLYEXPR\_ADD();

void VIEW\_POLYEXPR\_SUB();

};

POLYEXPR::POLYEXPR(int para)

{

order = para;

PE1 = NULL;

PE2 = NULL;

PE3 = NULL;

PE4 = NULL;

}

void POLYEXPR::READ\_POLYEXPR1()

{

NODE \*last = NULL;

cout<<endl<<"Enter poly Exp 1 : ";

for (int i=order;i>=0;i--)

{

NODE \*NN = new NODE();

cout<<endl<<"Enter Coeff of X^"<<i<<" : ";

cin>>NN->coeff;

NN->power = i;

if(PE1 == NULL)

PE1 = last = NN;

else

{

last->link = NN;

last = NN;

}

}

}

void POLYEXPR::READ\_POLYEXPR2()

{

NODE \*last = NULL;

cout<<endl<<"Enter poly Exp 2 : ";

for (int i=order;i>=0;i--)

{

NODE \*NN = new NODE();

cout<<endl<<"Enter Coeff of X^"<<i<<" : ";

cin>>NN->coeff;

NN->power = i;

if(PE2 == NULL)

PE2 = last = NN;

else

{

last->link = NN;

last = NN;

}

}

}

void POLYEXPR::ADD\_POLYEXPR()

{

NODE \*last = NULL;

NODE \*ptr1 =PE1;

NODE \*ptr2 =PE2;

cout<<endl<<"Addition of the given polynomial equations is : ";

for (int i=order;i>=0;i--)

{

NODE \*NN = new NODE();

// cout<<endl<<"Enter Coeff of X^"<<i<<" : ";

NN->coeff = ptr1->coeff + ptr2->coeff ;

NN->power = i;

if(PE3 == NULL)

PE3 = last = NN;

else

{

last->link = NN;

last = NN;

}

ptr1 = ptr1->link;

ptr2 = ptr2->link;

}

}

void POLYEXPR::SUB\_POLYEXPR()

{

NODE \*last = NULL;

NODE \*ptr1 =PE1;

NODE \*ptr2 =PE2;

cout<<endl<<"Addition of the given polynomial equations is : ";

for (int i=order;i>=0;i--)

{

NODE \*NN = new NODE();

NN->coeff = ptr1->coeff-ptr2->coeff ;

NN->power = i;

if(PE4 == NULL)

PE4 = last = NN;

else

{

last->link = NN;

last = NN;

}

ptr1 = ptr1->link;

ptr2 = ptr2->link;

}

}

void POLYEXPR::VIEW\_POLYEXPR\_ADD()

{

NODE \*ptr = PE3;

cout<<endl<<"Poly Exp 1 : ";

for (int i=order;i>=0;i--)

{

cout<<ptr->coeff<<"X^"<<i<<" + ";

ptr= ptr->link;

}

}

void POLYEXPR::VIEW\_POLYEXPR\_SUB()

{

NODE \*ptr = PE4;

cout<<endl<<"Poly Exp 2 : ";

for (int i=order;i>=0;i--)

{

cout<<ptr->coeff<<"X^"<<i<<" + ";

ptr= ptr->link;

}

}

void main()

{

int ord;

clrscr();

cout<<endl<<"Enter max order of Poly Expression : ";

cin>>ord;

POLYEXPR obj(ord);

obj.READ\_POLYEXPR1();

obj.READ\_POLYEXPR2();

obj.ADD\_POLYEXPR();

obj.VIEW\_POLYEXPR\_ADD();

obj.SUB\_POLYEXPR();

obj.VIEW\_POLYEXPR\_SUB();

getch();

}