adt.h

struct poly

{

    int coeff,exp;

};

struct polyADT

{

    struct poly p;

    struct polyADT \* next;

};

void insertEnd(struct polyADT \* header,int coeff,int exp);

struct polyADT polyAdd(struct polyADT\* p1,struct polyADT\* p2);

struct polyADT\*polyMul(struct polyADT\*p1,struct polyADT \*p2);

void display(struct polyADT\*header);

struct polyADT polySimplify(struct polyADT\*p);

void polyDegree(struct polyADT \*p);

int polyEvaluate(struct polyADT \*p);

impl.h

#include "adt.h"

#include<stdlib.h>

#include<stdio.h>

#include<math.h>

void insertEnd(struct polyADT\* header,int coeff,int exp)

{

    struct polyADT\*ptr,\*temp;

    ptr=(struct polyADT \*)malloc(sizeof(struct polyADT));

    ptr->p.coeff=coeff;

    ptr->p.exp=exp;

    temp=header;

    if(header->next!=NULL)

    {

        while(temp->next!=NULL)

            temp=temp->next;

        ptr->next=temp->next;

        temp->next=ptr;

    }

    else

    {

        ptr->next=header->next;

        header->next=ptr;

    }

}

void display(struct polyADT \* header)

{

    struct polyADT\*temp=header;

    temp=temp->next;

    while(temp!=NULL)

    {

        if(temp->next!=NULL)

        {

            if(temp->p.coeff<0)

                printf("%dx^%d ",temp->p.coeff,temp->p.exp);

            else

                printf("+%dx^%d ",temp->p.coeff,temp->p.exp);

        }

        else

        {

            if(temp->p.exp==0)

                if(temp->p.coeff<0)

                    printf("%d",temp->p.coeff);

                else

                    printf("+%d",temp->p.coeff);

            else

                if(temp->p.coeff<0)

                    printf("%dx^%d",temp->p.coeff,temp->p.exp);

                else

                    printf("+%dx^%d",temp->p.coeff,temp->p.exp);

        }

        temp=temp->next;

    }

    printf("\n");

}

struct polyADT polyAdd(struct polyADT\* p1,struct polyADT\* p2)

{

    struct polyADT \*p3;

    p3=(struct polyADT\*)malloc(sizeof(struct polyADT));

    p3->next=NULL;

    p1=p1->next;

    p2=p2->next;

    int f;

    while(p1!=NULL && p2!=NULL)

    {

        if(p1->p.exp>p2->p.exp)

        {

            insertEnd(p3,p1->p.coeff,p1->p.exp);

            p1=p1->next;

        }

        else if(p1->p.exp<p2->p.exp)

        {

            insertEnd(p3,p2->p.coeff,p2->p.exp);

            p2=p2->next;

        }

        else

        {

            insertEnd(p3,p1->p.coeff+p2->p.coeff,p2->p.exp);

            p1=p1->next;

            p2=p2->next;

        }

        if(p1==NULL && p2==NULL)

            f=0;

        else if(p1==NULL)

            f=1;

        else

            f=2;

    }

    if(f==1)

    {

        while(p2!=NULL)

        {

            insertEnd(p3,p2->p.coeff,p2->p.exp);

            p2=p2->next;

        }

    }

    else if(f==2)

    {

        while(p1!=NULL)

        {

            insertEnd(p3,p1->p.coeff,p1->p.exp);

            p1=p1->next;

        }

    }

    return \*p3;

}

struct polyADT\*polyMul(struct polyADT\*p1,struct polyADT \*p2)

{

    struct polyADT \*p3=(struct polyADT\*)malloc(sizeof(struct polyADT));

    p3->next=NULL;

    struct polyADT\*start=p2->next;

    p1=p1->next;

    while(p1!=NULL)

    {

        p2=start;

        while(p2!=NULL)

        {

            insertEnd(p3,p1->p.coeff\*p2->p.coeff,p1->p.exp+p2->p.exp);

            p2=p2->next;

        }

        p1=p1->next;

    }

    return p3;

}

struct polyADT polySimplify(struct polyADT\*p)

{

    struct polyADT\* back=p->next,\*front =back->next,\*temp,\*simp,\*start;

    simp=(struct polyADT\*)malloc(sizeof(struct polyADT));

    simp->next=NULL;

    start=simp->next;

    int f=0,sum\_co=0;

    while(back!=NULL)

    {

        f=0;

        start=simp;

        while(start!=NULL)

        {

            if(start->p.exp==back->p.exp)

            {

                f=1;

                break;

            }

            start=start->next;

        }

        if(f==0)

        {

            sum\_co=back->p.coeff;

            front =back->next;

            while(front!=NULL)

            {

                if(front->p.exp==back->p.exp)

                {

                    sum\_co+=front->p.coeff;

                }

                front=front->next;

            }

            insertEnd(simp,sum\_co,back->p.exp);

        }

        back=back->next;

    }

    return \*simp;

}

void polyDegree(struct polyADT \*p)

{

    printf("Degree of polynomial:%d\n",p->next->p.exp);

}

int polyEvaluate(struct polyADT \*p)

{

    int val;

    long int sum=0;

    printf("Enter value with which you need to evaluate the polynomial:");

    scanf("%d",&val);

    p=p->next;

    while(p!=NULL)

    {

        sum+=pow(val,p->p.exp)\*(p->p.coeff);

        p=p->next;

    }

    printf("Evaluated value:%d",sum);

}

appl.c

#include"impl.h"

#include<stdio.h>

int main()

{

    struct polyADT p1,p2,p3\_add,\*p3\_mul,simp;

    p1.next=NULL;

    p2.next=NULL;

    int coeff,exp,count,ch;

    printf("Enter count of terms of first polynomial:");

    scanf("%d",&count);

    for(int i=0;i<count;i++)

    {

        printf("Enter coefficient and exponent of term %d:",i+1);

        scanf("%d %d",&coeff,&exp);

        insertEnd(&p1,coeff,exp);

    }

    printf("Enter count of terms of second polynomial:");

    scanf("%d",&count);

    for(int i=0;i<count;i++)

    {

        printf("Enter coefficient and exponent of term %d:",i+1);

        scanf("%d %d",&coeff,&exp);

        insertEnd(&p2,coeff,exp);

    }

    printf("\nDisplaying polynomial 1:");

    display(&p1);

    printf("\nDisplaying polynomial 2:");

    display(&p2);

    p3\_add=polyAdd(&p1,&p2);

    printf("\nDisplaying sum of polynomials:");

    display(&p3\_add);

    p3\_mul=polyMul(&p1,&p2);

    printf("\nDisplaying product of polynomials:");

    display(p3\_mul);

    printf("\nSimplifying polynomial(if there is a need of combining terms):");

    simp=polySimplify(p3\_mul);

    display(&simp);

    printf("Degree of polynomial:\n");

    printf("Degree of which polynomial do you want? 1)First 2)Second 3)Sum 4)Product 5)Simplified version ");

    scanf("%d",&ch);

    if(ch==1)

        polyDegree(&p1);

    else if(ch==2)

        polyDegree(&p2);

    else if(ch==3)

        polyDegree(&p3\_add);

    else if(ch==4)

        polyDegree(p3\_mul);

    else

        polyDegree(&simp);

    printf("\nEvaluating polynomial:\n");

    printf("Which polynomial should be evaluated? 1)First 2)Second 3)Sum 4)Product 5)Simplified version ");

    scanf("%d",&ch);

    if(ch==1)

        polyEvaluate(&p1);

    else if(ch==2)

        polyEvaluate(&p2);

    else if(ch==3)

        polyEvaluate(&p3\_add);

    else if(ch==4)

        polyEvaluate(p3\_mul);

    else

        polyEvaluate(&simp);

}

**O/P:**

