```
//polyADTimpl.h
#include "polyADTif.h"
#include <stdlib.h>
#include <math.h>
void insertFront(polyADT *p,term t)//Input a polynomial through insertion at the front
{
	polyADT *temp;
	temp=malloc(sizeof(polyADT));
	temp->data=t;
	temp->next=p->next;
	p->next=temp;
}
void insertEnd(polyADT *p,term t)
{
	polyADT *ptr;
	polyADT *temp;
	temp=malloc(sizeof(polyADT));
	ptr=p;
	while(ptr->next!=NULL)
	{
		ptr=ptr->next;
	}
	ptr->next=temp;
	temp->next=NULL;
	temp->data=t;
}
void insertAfterTerm(polyADT *p, term t, int exp)
{
	polyADT *found,*temp;
```

```
        found=find(p,exp);
        if(found!=NULL)
        {
                temp=malloc(sizeof(polyADT));
                temp->data=t;
                temp->next=found->next;
                found->next=temp;
        }

}

polyADT *polyAdd(polyADT *p1, polyADT *p2)
{
        polyADT *poly1,*poly2,*result,*temp,*start;
        int flag=0;
        result=malloc(sizeof(polyADT));
        start=result;
        result->next=NULL;
        poly1=p1->next;//address of first node in first polynomial
        poly2=p2->next;//address of first node in second polynomial
        while(poly1!=NULL&&poly2!=NULL)
        {
                if(poly1->data.exp>poly2->data.exp)
                {
                        temp=malloc(sizeof(polyADT));
                        temp->data=poly1->data;
                        temp->next=NULL;
                        result->next=temp;
                        result=temp;
                poly1=poly1->next;
            }
            else if(poly2->data.exp>poly1->data.exp)
            {
                temp=malloc(sizeof(polyADT));
                temp->data=poly2->data;
                temp->next=NULL;
                result->next=temp;
                result=temp;
                poly2=poly2->next;
            }
            else
            {
                temp=malloc(sizeof(polyADT));
                temp->data.coeff=poly1->data.coeff+poly2->data.coeff;
                temp->data.exp=poly1->data.exp;
                temp->next=NULL;
                result->next=temp;
                result=temp;
                poly1=poly1->next;
                poly2=poly2->next;
            }
        }
            if(poly1!=NULL)//adds all remaining elements if poly 1 still has elements
            {
                while(poly1!=NULL)
                {
                    temp=malloc(sizeof(polyADT));
                    temp->data=poly1->data;
                    temp->next=NULL;
                    result->next=temp;
                    result=temp;
                    poly1=poly1->next;
                }
            }
            else
            {
                while(poly2!=NULL)//adds all remaining elements if poly 2 still has elements
                {
                    temp=malloc(sizeof(polyADT));
                    temp->data=poly2->data;
                    temp->next=NULL;
                    result->next=temp;
                    result=temp;
                    poly2=poly2->next;
                }

            }
            return start;
    }

    polyADT *mul(polyADT *p1,polyADT *p2)
    {//result is header node
        polyADT *poly1,*poly2,*result,*temp,*present,*curr;
        result=malloc(sizeof(polyADT));
        result->next=NULL;
        poly1=p1->next;//address of first node
        poly2=p2->next;//address of second node
```

```
	while(poly1!=NULL)
	{
		while(poly2!=NULL)
		{
			if(result->next==NULL)//checks if first node is going to be inserted
			{
				temp=malloc(sizeof(polyADT));//new node
				temp->data.coeff=poly1->data.coeff*poly2->data.coeff;
				temp->data.exp=poly1->data.exp+poly2->data.exp;
				temp->next=NULL;
				curr=temp;//curr has address of last node
				poly2=poly2->next;//moves to next term
				result->next=temp;//header now points to new node
			}
			else
			{
				present=find(result,poly1->data.exp+poly2->data.exp);//finds address
of exponent in result list
				if(present!=NULL)//exponent present
				{
					present->data.coeff+=poly1->data.coeff*poly2->data.coeff;
					poly2=poly2->next;
					//printf("Node updated\n");
				}
				else//exponent not present
				{
					temp=malloc(sizeof(polyADT));
					temp->data.coeff=poly1->data.coeff*poly2->data.coeff;
					temp->data.exp=poly1->data.exp+poly2->data.exp;
```

```
                        curr->next=temp;//last node now points to temp
                        temp->next=NULL;
                        curr=temp;//*last node changes now*
                        poly2=poly2->next;
                        //printf("new node");
                    }
                }
            }
            poly1=poly1->next;//moves
            poly2=p2;//again starts from the beginning of polynomial 2

        }
        return result;

}
polyADT *find(polyADT *list,int x)
{
        polyADT *temp_ptr;
        temp_ptr=list->next;
        while(temp_ptr!=NULL)
        {
            if(temp_ptr->data.exp==x)
                return temp_ptr;
            else
            {
                temp_ptr=temp_ptr->next;
            }
        }
        //printf("\nExponent not present!!\n");
        return NULL;
}
polyADT *polySimplify(polyADT *p)
{
        polyADT *result,*temp,*ptr,*start,*present,*final,*ptrnext,*ptrprev;
        int deg,currdeg,length=0,i;
        deg=polyDegree(p);
        int arr[100];
        ptr=p->next;
        while(ptr!=NULL)
        {
                ptrnext=ptr->next;
                ptrprev=ptr;
                while(ptrnext!=NULL)
                {
                        if(ptr->data.exp==ptrnext->data.exp)
                        {
                                ptr->data.coeff+=ptrnext->data.coeff;
                                ptrprev->next=ptrnext->next;
                        }
                        ptrprev=ptrprev->next;
                        ptrnext=ptrnext->next;
                }
                ptr=ptr->next;
        }

        ptr=p->next;
        while(ptr!=NULL)
        {
```

```
        length++;
        ptr=ptr->next;
}
//printf("%d",length);

result=malloc(sizeof(polyADT));
result->next=NULL;
start=result;

for(i=0;i<length;i++)
{
        ptr=p->next;
        while(ptr!=NULL)
        {
                if(ptr->data.exp==deg)
                {
                        //printf("\ndeg matches");
                        temp=malloc(sizeof(polyADT));
                        temp->data=ptr->data;
                        temp->next=result->next;
                        result->next=temp;
                        result=temp;
                }
                ptr=ptr->next;
        }
        deg--;
        //printf("%d",deg);
}
```

```
	return start;
}
int polyDegree(polyADT *p)
{
	polyADT *ptr;
	int deg;
	ptr=p->next;
	deg=0;
	while(ptr!=NULL)
	{
		if(ptr->data.exp>deg)
			deg=ptr->data.exp;
		ptr=ptr->next;
	}
	return deg;
}
int polyEvaluate(polyADT *p,int x)
{
	int result=0;
	polyADT *ptr;
	ptr=p->next;
	while(ptr!=NULL)
	{
		result+=ptr->data.coeff*(pow(x,ptr->data.exp));
		ptr=ptr->next;

	}
```

```
        return result;
}
//polyADTif.h
#include <stdio.h>
typedef struct
{
        int exp;
        int coeff;
}term;
struct polyADT
{
        term data;
        struct polyADT *next;
};

typedef struct polyADT polyADT;

void insertFront(polyADT *p,term t);//Input a polynomial through insertion at the front
void insertEnd(polyADT *p,term t);//Input a polynomial through insertion at the end
void insertAfterTerm(polyADT *p, term t, int exp);//Input a polynomial after a term
polyADT *polyAdd(polyADT *p1, polyADT *p2);//Add two polynomials
polyADT *polyMul(polyADT *p1, polyADT *p2);//multiply two polynomials
int polyDegree(polyADT *p);//Find the degree of polynomial
int polyEvaluate(polyADT *p,int x);//Evaluate a polynomial
polyADT *polySimplify(polyADT *p);//Simplifying the polynomial – Combining like terms
polyADT *find(polyADT *list,int x);//find a given exponent in a polynomial

//polyADTappl.c
#include "polyADTimpl.h"
```

```
term termInput();
void polyInput(polyADT *p);
void display(polyADT *poly);
void main()
{
	int ch1=1,ch=1,deg,x,result;
	polyADT *p1,*p2,*res;
	term t;
	printf("\n\nEnter the desired option\n1 to add\n2 to multiply\n3 for degree of polynomial\n4 to
evaluate\n5 to exit\nYour choice: ");
	scanf("%d",&ch);
	do
	{
		switch(ch)
		{
			case 1:
				printf("\nEnter the data for the 2 polynomials to add:\n");
				p1=malloc(sizeof(polyADT));
				p1->next=NULL;
				printf("\nPOLYNOMIAL 1\n");
				polyInput(p1);
				printf("\npolynomial entered:\n");
				display(p1);
				p1=polySimplify(p1);
				printf("\n\nsimplified polynomial\n");
				display(p1);
				p2=malloc(sizeof(polyADT));
				p2->next=NULL;
				printf("\nPOLYNOMIAL 2\n\n");
```

```
        polyInput(p2);
        printf("\npolynomial entered:\n");
        display(p2);
        p2=polySimplify(p2);
        printf("\n\nsimplified polynomial\n");
        display(p2);
        res=polyAdd(p1,p2);
        res=polySimplify(res);
        printf("\n\nThe resultant is:\n");
        display(res);
        break;
case 2:
        printf("Enter the data for the 2 polynomials to multiply:\n");
        p1=malloc(sizeof(polyADT));
        p1->next=NULL;
        printf("\nPOLYNOMIAL 1\n");
        polyInput(p1);
        printf("polynomial entered:\n");
        display(p1);
        p1=polySimplify(p1);
        printf("\n\nsimplified polynomial\n");
        display(p1);
        p2=malloc(sizeof(polyADT));
        p2->next=NULL;
        printf("\nPOLYNOMIAL 2\n");
        polyInput(p2);
        printf("polynomial entered:\n");
        display(p2);
        p2=polySimplify(p2);
```

```
                printf("\n\nsimplified polynomial\n");
                display(p2);
                res=mul(p1,p2);
                res=polySimplify(res);
                printf("\n\nThe resultant is:\n");
                display(res);
                break;
            case 3:
                printf("\nEnter the polynomial for which you want to find the
degree:\n");
                p1=malloc(sizeof(polyADT));
                p1->next=NULL;
                printf("\nPOLYNOMIAL \n");
                polyInput(p1);
                printf("\npolynomial entered:\n");
                display(p1);
                p1=polySimplify(p1);
                printf("\n\nsimplified polynomial\n");
                display(p1);
                deg=polyDegree(p1);
                printf("\nThe degree of the polynomial is: %d",deg);
                break;
            case 4:
                printf("\nEnter the polynomial: \n");
                p1=malloc(sizeof(polyADT));
                p1->next=NULL;
                printf("\nPOLYNOMIAL \n");
                polyInput(p1);
                p1=polySimplify(p1);
```

```
                                        display(p1);

                                        printf("\nEnter the value to evaluate the polynomial with: ");

                                        scanf("%d",&x);

                                        result=polyEvaluate(p1,x);

                                        printf("\nThe result is: %d",result);

                                        break;

                    }

                    printf("\n\nEnter the desired option\n1 to add\n2 to multiply\n3 for degree of
polynomial\n4 to evaluate\n5 to exit\nYour choice: ");

                    scanf("%d",&ch);

          }while(ch!=5);




}

void display(polyADT *poly)

{

          polyADT *ptr;

          ptr=poly->next;

          while(ptr!=NULL)

          {

                    if(ptr->next==NULL)

                              printf("%d x^%d",ptr->data.coeff,ptr->data.exp);

                    else

                              printf("%d x^%d + ",ptr->data.coeff,ptr->data.exp);

                    ptr=ptr->next;

          }

}

term termInput()
```

```
{
        term t;
        printf("Enter the coefficient: ");
        scanf("%d",&t.coeff);
        printf("Enter the exponent: ");
        scanf("%d",&t.exp);
        return t;
}
void polyInput(polyADT *p)
{
        int ch1,exp;
        term t;
        printf("\nEnter the desired option\n1 to insert in the front\n2 to insert after an exponent\n3 to
insert at the end \n4 to stop entering\nyour choice:    ");
        scanf("%d",&ch1);
        while(ch1!=4)
        {
                switch(ch1)
                {
                        case 1:
                                t=termInput();
                                insertFront(p,t);
                                break;
                        case 2:
                                t=termInput();
                                printf("Enter the exponent after which you want to insert: ");
                                scanf("%d",&exp);
                                insertAfterTerm(p,t,exp);
                                break;
```

```
				case 3:

					t=termInput();

					insertEnd(p,t);

					break;

			}

			printf("\nEnter the desired option\n1 to insert in the front\n2 to insert after an 
exponent\n3 to insert at the end \n4 to stop entering\nyour choice:    ");

			scanf("%d",&ch1);

		}

}


/*

OUTPUT:



Enter the desired option

1 to add

2 to multiply

3 for degree of polynomial

4 to evaluate

5 to exit

Your choice: 1


Enter the data for the 2 polynomials to add:


POLYNOMIAL 1


Enter the desired option

1 to insert in the front
```

```
2 to insert after an exponent
3 to insert at the end
4 to stop entering
your choice:    1
Enter the coefficient: 3
Enter the exponent: 2

Enter the desired option
1 to insert in the front
2 to insert after an exponent
3 to insert at the end
4 to stop entering
your choice:    1
Enter the coefficient: 3
Enter the exponent: 1

Enter the desired option
1 to insert in the front
2 to insert after an exponent
3 to insert at the end
4 to stop entering
your choice:    1
Enter the coefficient: 3
Enter the exponent: 0

Enter the desired option
1 to insert in the front
2 to insert after an exponent
3 to insert at the end
```

```
4 to stop entering

your choice:    4


polynomial entered:

3 x^0 + 3 x^1 + 3 x^2


simplified polynomial

3 x^2 + 3 x^1 + 3 x^0

POLYNOMIAL 2



Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:    1

Enter the coefficient: 4

Enter the exponent: 4


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:    1

Enter the coefficient: 4

Enter the exponent: 3
Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:    1

Enter the coefficient: 4

Enter the exponent: 2


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:    4


polynomial entered:

4 x^2 + 4 x^3 + 4 x^4


simplified polynomial

4 x^4 + 4 x^3 + 4 x^2


The resultant is:

4 x^4 + 4 x^3 + 7 x^2 + 3 x^1 + 3 x^0


Enter the desired option

1 to add

2 to multiply

3 for degree of polynomial
```

```
4 to evaluate

5 to exit

Your choice: 3


Enter the polynomial for which you want to find the degree:


POLYNOMIAL


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:     1

Enter the coefficient: 3

Enter the exponent: 4


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:     2

Enter the coefficient: 4

Enter the exponent: 3

Enter the exponent after which you want to insert: 4


Enter the desired option

1 to insert in the front
```

```
2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:     4


polynomial entered:

3 x^4 + 4 x^3


simplified polynomial

3 x^4 + 4 x^3

The degree of the polynomial is: 4


Enter the desired option

1 to add

2 to multiply

3 for degree of polynomial

4 to evaluate

5 to exit

Your choice: 4


Enter the polynomial:


POLYNOMIAL


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering
```

```
your choice:    1

Enter the coefficient: 3

Enter the exponent: 3


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:    3

Enter the coefficient: 3

Enter the exponent: 4


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:    4

3 x^4 + 3 x^3

Enter the value to evaluate the polynomial with: 3


The result is: 324


Enter the desired option

1 to add

2 to multiply

3 for degree of polynomial

4 to evaluate
```

```
5 to exit

Your choice: 2

Enter the data for the 2 polynomials to multiply:


POLYNOMIAL 1


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:    1

Enter the coefficient: 1

Enter the exponent: 1


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering

your choice:    3

Enter the coefficient: 2

Enter the exponent: 2


Enter the desired option

1 to insert in the front

2 to insert after an exponent

3 to insert at the end

4 to stop entering
```

```
your choice:    3
Enter the coefficient: 2
Enter the exponent: 3

Enter the desired option
1 to insert in the front
2 to insert after an exponent
3 to insert at the end
4 to stop entering
your choice:    4
polynomial entered:
1 x^1 + 2 x^2 + 2 x^3

simplified polynomial
2 x^3 + 2 x^2 + 1 x^1
POLYNOMIAL 2

Enter the desired option
1 to insert in the front
2 to insert after an exponent
3 to insert at the end
4 to stop entering
your choice:    4
polynomial entered:


simplified polynomial
```

```
The resultant is:

0 x^2 + 0 x^1


Enter the desired option

1 to add

2 to multiply

3 for degree of polynomial

4 to evaluate

5 to exit

Your choice: 5

*/
```