SSN COLLEGE OF ENGINEERING (Autonomous)

Affiliated to Anna University

DEPARTMENT OF CSE

UCS308 Data Structures Lab

Assignment 3

Polynomial Manipulation

Register Number : 185001131

Name : Sai Charan B

Class : CSE – B


```
#include<stdio.h>
#include<stdlib.h>
typedef struct mynode
{     int coeff;
      int expo;
      struct mynode *next;
}node;
void display (node*);
void del(node*);
void sort(node*);
node* Create()
{     node *h;
      h=(node*)malloc(sizeof(node));
      h->next=NULL;
      return h;
}
void insert(node *a)
{     printf("\nEnter coeff and exponent\n");
      int c,e,op,f;
      scanf("%d",&c);
      node *p,*t;
      t=a;
      scanf("%d",&e);
      p=(node*)malloc(sizeof(node));
      p->coeff=c;
      p->expo=e;
      while(t->next!=NULL)
```

```
            t=t->next;
      p->next=t->next;
      t->next=p;
      display(a);
      printf("\nPress 0 to quit, 1 to continue\n");
      scanf("%d",&op);
      if(op)
            insert(a);


}
void display(node *a)
{     node *t;
      int c,e;
      t=a->next;
      while(t!=NULL)
      {     c=t->coeff;
            e=t->expo;
            if(t->next==NULL)
            {     printf(" %dX^%d\n",c,e);
            }
            else
            {     printf(" %dX^%d +",c,e);
            }
            t=t->next;
      }
}
void ins(node *a, int c,int e)
{     node *p,*t;
      p =(node*)malloc(sizeof(node));
      p->expo = e;
      p->coeff=c;
      t = a;
      while(t->next!=NULL)
            t = t->next;
      p->next =t->next;
      t->next = p;
}
node* add(node *p1,node *p2)
{
      node *p,*q,*ptr3;
      ptr3=(node*)malloc(sizeof(node));
      ptr3->next = NULL;
      p=(node*)malloc(sizeof(node));
      q=(node*)malloc(sizeof(node));
      p=p1->next;
      q=p2->next;
      int c,e;
      while(p!=NULL && q!=NULL)
      {
            if(p->expo>q->expo)
            {     c=p->coeff;
                  e=p->expo;
                  ins(ptr3,c,e);
                  p=p->next;
            }
            else if(p->expo<q->expo)
            {     c=q->coeff;
```

```
                e=q->expo;
                ins(ptr3,c,e);
                q=q->next;
            }
            else
            {
                if(p->coeff+q->coeff!=0)
                {   c=(p->coeff+q->coeff);
                    e=p->expo;
                    ins(ptr3,c,e);
                }
                p=p->next;
                q=q->next;
            }
        }
        while(p!=NULL)
        {
            ins(ptr3,p->coeff,p->expo);
            p=p->next;
        }
        while(q!=NULL)
        {
            ins(ptr3,q->coeff,q->expo);
            q=q->next;
        }
        return ptr3;
}
node* mult(node *a,node *b)
{   node *p,*q,*r;
        r=(node*)malloc(sizeof(node));
        p=a->next;
        q=b->next;
        int c,e;
        while(p!=NULL)
        {   while(q!=NULL)
            {   c=(p->coeff*q->coeff);
                e=(p->expo+q->expo);
                ins(r,c,e);
                q=q->next;
            }
            p=p->next;
            q=b->next;
        }
        del(r);
        sort(r);
        return(r);
}
void sort(node*a)
{   node *p,*q;
        int c,e;
        p=a->next;
        while(p!=NULL)
        {   q=p->next;
            while(q!=NULL)
            {   if(q->expo>p->expo)
                {   c=p->coeff;
                    e=p->expo;
                    p->expo=q->expo;
```

```
                        p->coeff=q->coeff;
                        q->expo=e;
                        q->coeff=c;
                    }
                    q=q->next;
                }
                p=p->next;

        }
}
void del(node* start)
{
    node *ptr1, *ptr2, *dup;
    ptr1 = start;


    while (ptr1 != NULL && ptr1->next != NULL) {
        ptr2 = ptr1;


        while (ptr2->next != NULL) {


            if (ptr1->expo == ptr2->next->expo) {


                ptr1->coeff = ptr1->coeff + ptr2->next->coeff;
                dup = ptr2->next;
                ptr2->next = ptr2->next->next;
                free(dup);
            }
            else
                ptr2 = ptr2->next;
        }
        ptr1 = ptr1->next;
    }
}
void main()
{     node *a=Create();
      insert(a);
      display(a);
      node *b=Create();
      insert(b);
      display(b);
      node *c=Create();
      c=add(a,b);
      printf("\n\nsum");
      display(c);
      node *d=Create();
      d=mult(a,b);
      printf("\n\nproduct");
      display(d);
}
```

Output:

```
Enter coeff and exponent
3
12
 3X^12

Press 0 to quit, 1 to continue
1

Enter coeff and exponent
8
8
 3X^12 + 8X^8

Press 0 to quit, 1 to continue
1

Enter coeff and exponent
-22
4
 3X^12 + 8X^8 + -22X^4

Press 0 to quit, 1 to continue
1

Enter coeff and exponent
3
1
 3X^12 + 8X^8 + -22X^4 + 3X^1

Press 0 to quit, 1 to continue
1
```

```
Enter coeff and exponent
-7
0
 3X^12 + 8X^8 + -22X^4 + 3X^1 + -7X^0

Press 0 to quit, 1 to continue
0
 3X^12 + 8X^8 + -22X^4 + 3X^1 + -7X^0

Enter coeff and exponent
7
14
 7X^14

Press 0 to quit, 1 to continue
1

Enter coeff and exponent
-10
9
 7X^14 + -10X^9

Press 0 to quit, 1 to continue
1

Enter coeff and exponent
-8
8
 7X^14 + -10X^9 + -8X^8

Press 0 to quit, 1 to continue
1
```

```
Enter coeff and exponent
6
5
 7X^14 + -10X^9 + -8X^8 + 6X^5

Press 0 to quit, 1 to continue
1

Enter coeff and exponent
-9
1
 7X^14 + -10X^9 + -8X^8 + 6X^5 + -9X^1

Press 0 to quit, 1 to continue
0
 7X^14 + -10X^9 + -8X^8 + 6X^5 + -9X^1



sum 7X^14 + 3X^12 + -10X^9 + 6X^5 + -22X^4 + -6X^1 + -7X^0



product 21X^26 + 56X^22 + -30X^21 + -24X^20 + -154X^18 + -62X^17 + -
64X^16 + 21X^15 + -49X^14 + 241X^13 + 176X^12 + -30X^10 + -158X^9 +
56X^8 + 18X^6 + 156X^5 + -27X^2 + 63X^1
```