Assignment - 1

[Submitted by Jaya Gupta(200471) and Pratyush Gupta(200717)]

Some Basic Function

```
struct node:
    int coeff,exp
    node* next,prev

//creates a new node

function make_node(c,e):
    temp=new node()
    temp.coeff=c
    temp.exp=e
    temp.prev=temp.next=null
    return temp

//append node at the end of the list
function append(c,sentinel):
    node.prev=sentinel.prev
    node.next=sentinel
    sentinel.prev.next=node
    sentinel.prev=node

//used for taking input [p(x) and q(x) pol]
function input(n):
    sentinel=new node()
    sentinel.coeff=sentinel.exp=0
    sentinel.prev=sentinel.next=sentinel

    i=0
    while(i<n) do:
        cin>>coeff>>exp
        c=make_node(coeff,exp)
        append(c,sentinel)

    return sentinel

//print function
function print(head):
    curr=head.next
```

```
    while(curr != head) do:
        cout<<curr.coeff<<curr.exp
        curr=curr.next
```

Main Driver Code

1.)

```
function addition(p,q,ansList):
    node *c1=p.next, *c2=q.next        //c1 and c2 starts from the first
node of respective list

    while(c1 !=p && c2!=q) do://Iterate until any one of the list finishes
        if(c1.exp<c2.exp):
//The node with the smaller exponent is appended
            node temp=make_node(c1.coeff,c1.exp)
            append(temp,ansList)
            c1=c1.next
 //The pointer with small exponent in advanced.

        else if(c2.exp<c1.exp):
            node temp=make_node(c2.coeff,c2.exp)
            append(c2,ansList)
            c2=c2.next
        Else: //If equal then the coeff are added and both pointers advance
            node temp=make_node(c1.coeff+c2.coeff,c2.exp)
            if(temp.coeff !=0) append(temp,ansList)
            c1=c1.next
            c2=c2.next

    node *c= c1==p?c2:c1
    while(c!=p && c!=q) do:
        node temp=make_node(c.coeff,c.exp)
        append(temp,ansList) //If one of the lists is done, then the
        c=c.next             //remaining can simply be appended.
    return ansList

//main function
n,m
cin>>n>>m
p=input(n)
q=input(m)       //p and q points toward sentinel node of respective list
```

```
ansList=addition(p,q) //ans list points toward sentinel node of final ans
list

print(ansList)
```

Time Complexity Analysis:

Suppose the length of p(x) is n and length of q(x) is m..

The ‘append’ operation takes place in constant time.

We follow the two pointers approach. The worst case for this algorithm would be when no exponent in one of the polynomials will match with an exponent in the other. So at every iteration only one of the pointers will advance. Since there are a total of m+n elements there can be utmost m+n iterations.

In every iteration, the operations that are performed take only constant time. And regardless of the case n an iteration, the time taken by the operations can be said to be bounded by a constant C.

So total time is less than or equal to C(m+n).

So complexity = O(m+n)