```
/*
  Author : Abhishek Yadav
  Roll Number: CS12B032

   this programs multiplies to large numbers (say a 1 million digit number with another
   large number and prints its output.
   implemetation has been done using linked list.
*/



#include<iostream>
#include<cmath>
#include<string>
#include<cstring>
#include"double_linked_list.cpp"
using namespace std;


list* multiply(list* num_1, list* num_2)
{
   if(num_1==NULL || num_2==NULL){
   cout <<"enter both numbers properly:\n";
   return NULL;
   }
   list *num1=num_1;
   list* num2=num_2;
   list* result=NULL;
   list* temp=NULL;
   list* pos_shift=NULL;
    list*traverse=NULL;
     while(num2!=NULL){
       int carry=0,result_digit=0;;
       int k=num2->data;
       temp=num1;
       if(result !=NULL){
          traverse=pos_shift;
          while(temp!=NULL){
          if(traverse!=NULL){
              result_digit=k*temp->data+traverse->data+carry;
              traverse->data=result_digit%10;
              carry=result_digit/10;
              traverse=traverse->rear;
              }
                else{
                result_digit=k*temp->data+carry;
               result=add_node_in_list(result,result_digit%10);
                carry=result_digit/10;
               }
                           temp=temp->rear;
          }
          pos_shift=pos_shift->rear;
```

```
        }
        else if(result==NULL){
        while(temp!=NULL){
          result_digit=temp->data*k+carry;
          result=add_node_in_list(result,result_digit%10);
          carry=result_digit/10;
          temp=temp->rear;
        }
        pos_shift=result->rear;
      }
        while(carry>0){
         result=add_node_in_list(result,carry%10);
         carry/=10;
        }
        num2=num2->rear;
     }
   return result;

}
int main()
{     int flag1=0,flag2=0;
      cout <<"Enter first Number\n";
      string str;
      cin>>str;
      if(str[0]=='-')
      flag1++;
      list*long_num1=string_to_int(str);
      cout <<"Now enter second number:\n";
      cin >>str;
      if(str[0]=='-')
      flag2++;
      list* long_num2=string_to_int(str);
     cout <<"result is : \n";
     if(flag1^flag2)
     cout <<"-";
         list*traverse=NULL;
     int l1=list_size(long_num1);
     int l2=list_size(long_num2);

       list *l_num1= l1>=l2 ?long_num1 : long_num2;
       list*l_num2= l1<l2 ?long_num1: long_num2;
     traverse= multiply(l_num1,l_num2);

 while(traverse!=NULL){
    cout <<traverse->data ;
    traverse=traverse->front;
   }
    cout <<endl;

       return 0;
 }
```

```
 ////////////////////////////

 //Library for double linked list

 using namespace std;
typedef struct linked_list list;
struct linked_list{
       list* rear;
       int data;
       list* front;
};
typedef struct list_ptr ptrs;
struct list_ptr{
    list* lst1;
    list* lst2;
};


list* list_new(int num){
    list *new_node=new list;
     new_node->rear=NULL;
     new_node->data=num;
     new_node->front=NULL;
    return new_node;
}

list *add_node_in_list(list*l,int data_element){
     list*flag_node=l;
     list*new_node=list_new(data_element);
          if(l==NULL)
        l=new_node;
     else{
          while(flag_node->rear!=NULL)
          flag_node=flag_node->rear;

          new_node->front=flag_node;
          flag_node->rear=new_node;
     }
     return l;
}

list * add_list_in_reverse(list* l,int data_element)
{
    list*flag_node=l;
     list *new_node=new list;
     new_node->rear=NULL;
     new_node->data=data_element;
     new_node->front=NULL;
     if(l!=NULL){
          new_node->rear=flag_node;
          flag_node->front=new_node;
          l=new_node;
```

```
    }
    else if(l==NULL)
      l=new_node;
      return l;

}

int list_size(list*l)
{
  list*temp_list=l;
  int counter=0;
  while(temp_list!=NULL)
  {
     temp_list=temp_list->rear;
     counter++;
  }
  return counter;
}
list* string_to_int(string& str)
{
  int l=str.length();
  list* long_num1=NULL ;
  int p,i=0;
  while(str[i]=='0' && i<l)
  i++;
  if(i==l){
  long_num1=add_node_in_list(long_num1,0);
  return long_num1;
  }else{
  while(i<l){
   if(isdigit(str[i])){
      p=str[i++]-'0';
      long_num1=add_list_in_reverse(long_num1,p);
      }else if(str[0]=='-')
          i++;
      else break;
  }
 }
  if(i<l)
  return NULL;
  else return long_num1;

}
list**string_to_int_for_division(string &str){
        int len=str.length();
  list** l=new list* [2];
  l[0]=NULL;
  l[1]=NULL;
  int p,i=0;
  while(str[i]=='0' && i<len)
  i++;
```

```
  if(i==len){
   l[0]=add_node_in_list(l[0],0);
   l[1]=add_list_in_reverse(l[1],0);
   return l;
  }else{
  int temp=len-1;
  int q;
  while(i<len){
    if(isdigit(str[i])){
       p=str[i++]-'0';
       q=str[temp--]-'0';
        l[0]=add_node_in_list(l[0],p);
        l[1]=add_list_in_reverse(l[1],q);
       }else if(str[0]=='-')
           i++;
       else break;
  }
 }
  if(i<len)
  return NULL;
  else return l;

}
```