DATA STRUTCURES FILE

Roll No.-2019UCO1580
Name-HARSHIT GUPTA
Course- CECSC02
Teacher- Mr. Rajeev Kumar

आ नो भद्रा क्रतवो यन्तु विश्वतः

NSUT

NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

INDEX

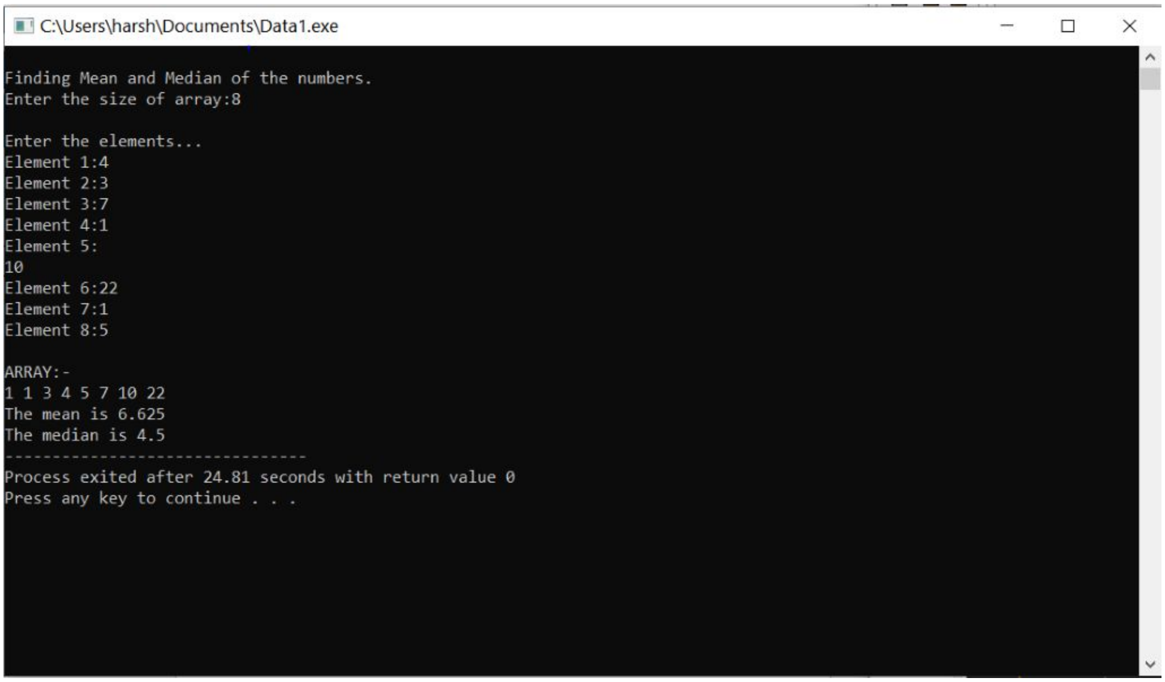
1. WAP to find the mean and the median of the numbers stored in the array

- **C++ code**

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   cout<<"\nFinding Mean and Median of the numbers.\n";
     int n;
   cout<<"Enter the size of array:";
   cin>>n;
     float arr[n];
     float sum=0;
   cout<<"\nEnter the elements...\n";
   for(int i=0;i<n;i+=1)
     {
      cout<<"Element "<<i+1<<":";
      cin>>arr[i];
     sum+=arr[i];
     }
     sort(arr,arr+n);
     float mean=sum/n;
   cout<<"\nARRAY:- \n";
   for(int i=0;i<n;i+=1)
      cout<<arr[i]<<" ";
   cout<<"\nThe mean is "<<mean;
     if(n%2==1)
     {
```

```
    int index= (n+1)/2;
     cout<<"\nThe median is "<<arr[index-1];
    }
    else
    {
    int index1=n/2;
    int index2=index1+1;
     cout<<"\nThe median is
"<<(arr[index1-1]+arr[index2-1])/2;
    }
    return 0;
}
```

- **OUTPUT**



```
Finding Mean and Median of the numbers.
Enter the size of array:8

Enter the elements...
Element 1:4
Element 2:3
Element 3:7
Element 4:1
Element 5:
10
Element 6:22
Element 7:1
Element 8:5

ARRAY:-
1 1 3 4 5 7 10 22
The mean is 6.625
The median is 4.5
--------------------------------
Process exited after 24.81 seconds with return value 0
Press any key to continue . . .
```

2.WAP to insert one element in an array and delete an element from an array

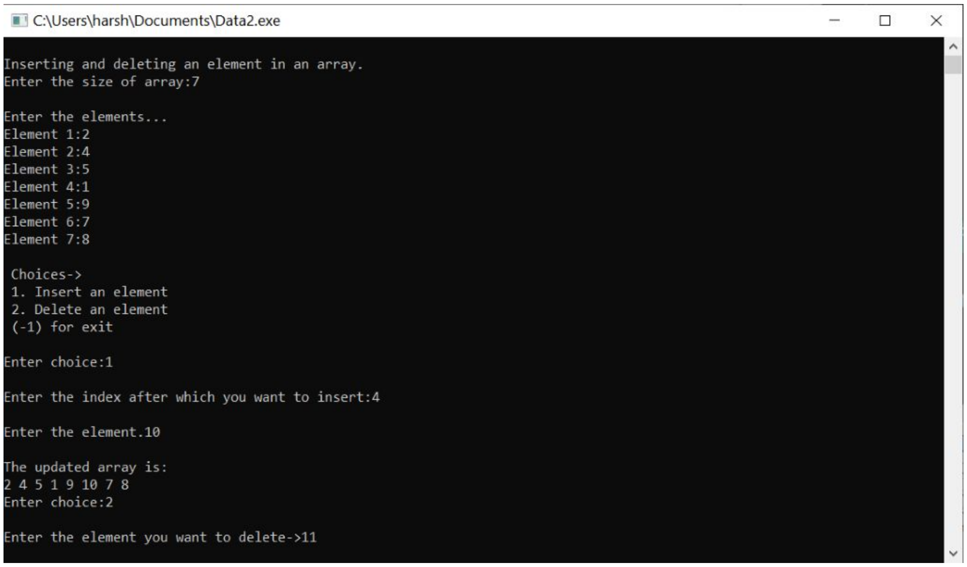
- **C++ code**

```
#include<bits/stdc++.h>
using namespace std;
int main()
{    cout<<"\nInserting and deleting an element in an
array.\n";
     int n;
   cout<<"Enter the size of array:";
   cin>>n;
     float arr[100000];
   cout<<"\nEnter the elements...\n";
   for(int i=0;i<n;i+=1)
     {
      cout<<"Element "<<i+1<<":";
      cin>>arr[i];
     }
   cout<<"\n Choices-> \n 1. Insert an element\n 2. Delete an element\n (-1) for exit \n";
     int choice;
   while(true )
   {    cout<<"\nEnter choice:";
   cin>>choice;
     if(choice==1)
     {
     int index,ele;
      x:cout<<"\nEnter the index after which you want to insert:";
      cin>>index;
     if(index>n)
          { cout<<"\nIvalid index.\nEnter again.";
          goto x;}
      cout<<"\nEnter the element.";
      cin>>ele;
      for(int i=n-1;i>index;i-=1)
     { arr[i+1]=arr[i];
     }
```

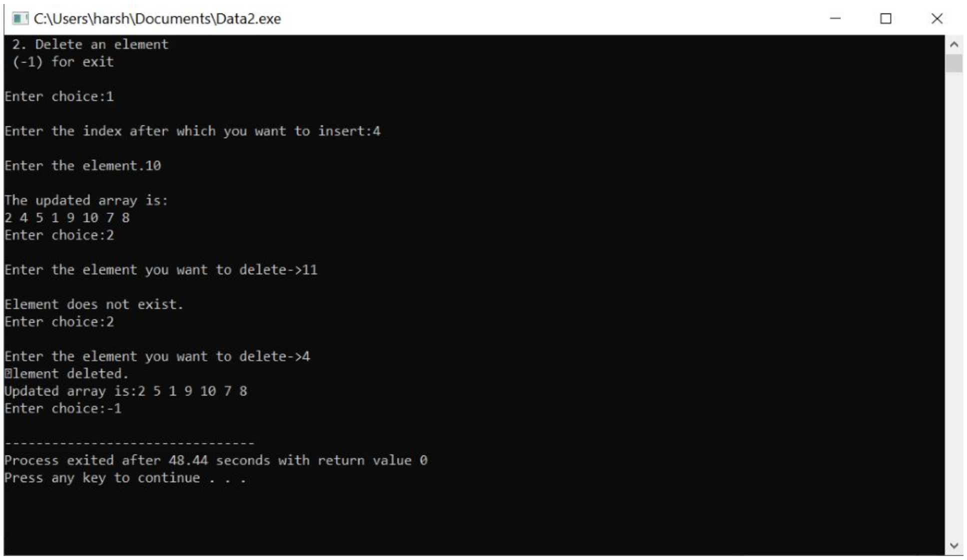
```
        arr[index+1]=ele;
       n++;
        cout<<"\nThe updated array is:\n";
        for(int i=0;i<n;i+=1)
        {    cout<<arr[i]<<" ";
       }
       }
       if(choice==2)
       {
        cout<<"\nEnter the element you want to
delete->";
       int del;
       bool flag= false;
        cin>>del;
       int index;
        for(int i=0;i<n;i+=1)
        { if(arr[i]==del)
             {
             index=i;
             flag=true;
             break;
             }
       }
       if(flag==false) cout<<"\nElement does not
exist.";
```

```
    else
    {
        for(int i=index;i<n-1;i+=1)
         { arr[i]=arr[i+1];
         }
         n--;
        cout<<"\Element deleted.\nUpdated array
is:";
        for(int i=0;i<n;i+=1)
           cout<<arr[i]<<" ";
    }
    }
    if(choice==-1) break;
}return 0;
}
```

- **OUTPUT**



```
Inserting and deleting an element in an array.
Enter the size of array:7

Enter the elements...
Element 1:2
Element 2:4
Element 3:5
Element 4:1
Element 5:9
Element 6:7
Element 7:8

 Choices->
 1. Insert an element
 2. Delete an element
 (-1) for exit

Enter choice:1

Enter the index after which you want to insert:4

Enter the element.10

The updated array is:
2 4 5 1 9 10 7 8
Enter choice:2

Enter the element you want to delete->11
```



```
 2. Delete an element
 (-1) for exit

Enter choice:1

Enter the index after which you want to insert:4

Enter the element.10

The updated array is:
2 4 5 1 9 10 7 8
Enter choice:2

Enter the element you want to delete->11

Element does not exist.
Enter choice:2

Enter the element you want to delete->4
□lement deleted.
Updated array is:2 5 1 9 10 7 8
Enter choice:-1

--------------------------------
Process exited after 48.44 seconds with return value 0
Press any key to continue . . .
```

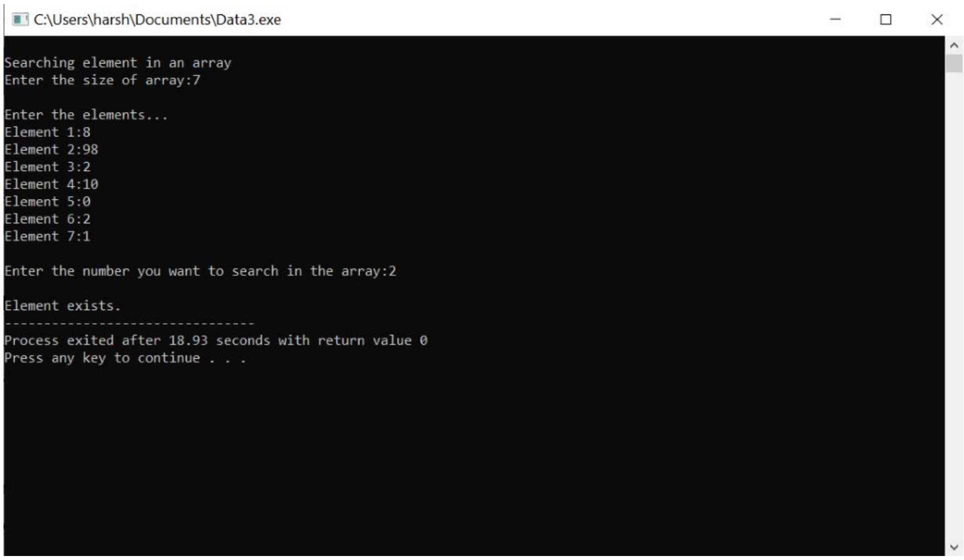
3. WAP to search for a number in an array

- **C++ CODE**

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   cout<<"\nSearching element in an array\n";
      int n;
   cout<<"Enter the size of array:";
   cin>>n;
      float arr[100000];
   cout<<"\nEnter the elements...\n";
   for(int i=0;i<n;i+=1)
      {
       cout<<"Element "<<i+1<<":";
       cin>>arr[i];
      }
   cout<<"\nEnter the number you want to search in the
array:";
      int ele;
      bool found=false;
   cin>>ele;
   for(int i=0;i<n;i+=1)
   {        if(ele==arr[i])
      {found=true;
      break;
```

```
        }
    }   if(found) cout<<"\nElement exists.";
        else cout<<"\nElement does not exist.";
        return 0;
}
```

- **OUTPUT**



```
Searching element in an array
Enter the size of array:7

Enter the elements...
Element 1:8
Element 2:98
Element 3:2
Element 4:10
Element 5:0
Element 6:2
Element 7:1

Enter the number you want to search in the array:2

Element exists.
--------------------------------
Process exited after 18.93 seconds with return value 0
Press any key to continue . . .
```

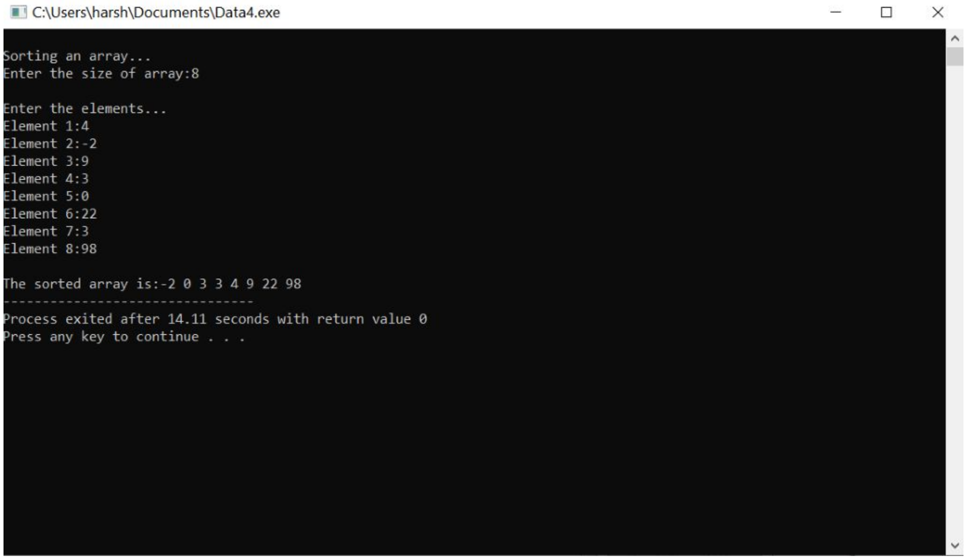
4. WAP to sort an array

- **C++ CODE**

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    cout<<"\nSorting an array...\n";
       int n;
    cout<<"Enter the size of array:";
    cin>>n;
       float a[100000];
    cout<<"\nEnter the elements...\n";
    for(int i=0;i<n;i+=1)
       {
        cout<<"Element "<<i+1<<":";
        cin>>a[i];
       }
       int min,minind,temp;
    for(int i=0;i<n;i+=1)
    {    min=a[i];
        minind=i;
        for(int j=i+1;j<n;j+=1)
       {
              if(a[j]<min)
              {min=a[j];
                minind=j;}
       }
```

```
        if(minind!=i)
        {
                temp=a[i];
                a[i]=a[minind];
                a[minind]=temp;
        }
        }    cout<<"\nThe sorted array is:";
      for(int i=0;i<n;i+=1)
         cout<<a[i]<<" ";
        return 0;
}
```

- **OUTPUT**



```
Sorting an array...
Enter the size of array:8

Enter the elements...
Element 1:4
Element 2:-2
Element 3:9
Element 4:3
Element 5:0
Element 6:22
Element 7:3
Element 8:98

The sorted array is:-2 0 3 3 4 9 22 98
--------------------------------
Process exited after 14.11 seconds with return value 0
Press any key to continue . . .
```

5. WAP to merge two sorted arrays

- **C++ CODE**

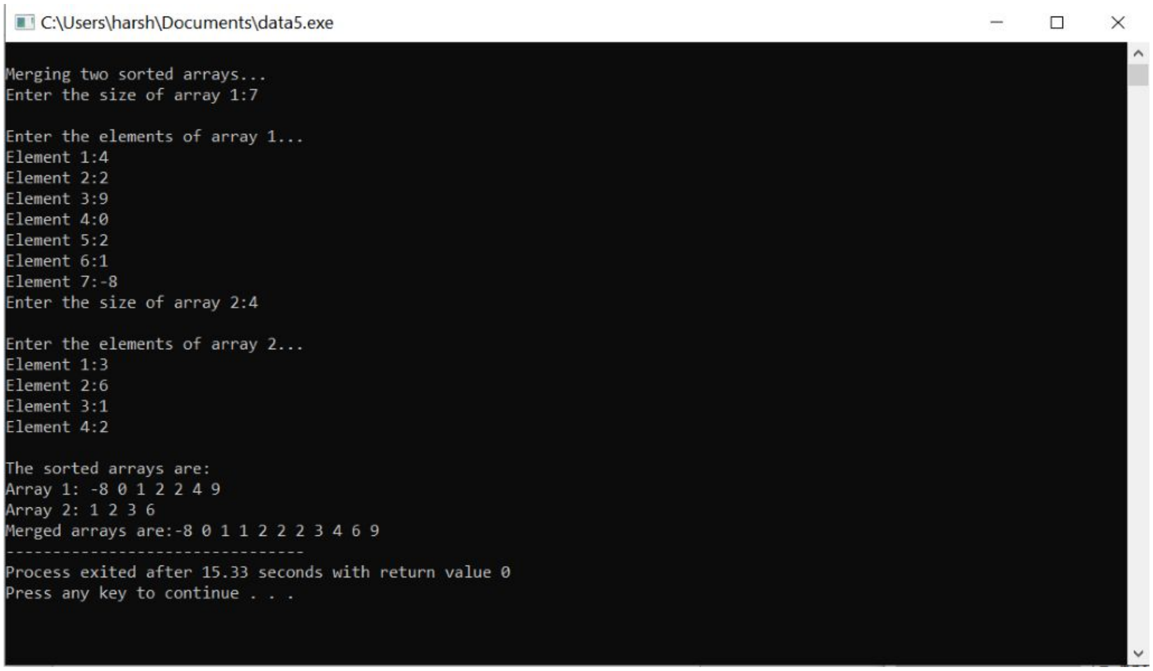
```
#include<bits/stdc++.h>
#define For(i,n) for(int i=0;i<n;i++)
using namespace std;
void merged(float a[],int n,float b[], int m)
{    cout<<"\nMerged arrays are:";
     int k,i,j;
   i=j=k=0;
     while(k<n+m)
     {
     if(a[i]<b[j] || j==m)
          { cout<<a[i]<<" ";
          i+=1;
          k+=1;}
     else if(a[i]>b[j] || i==n)
          {
            cout<<b[j]<<" ";
          j+=1;
          k+=1;
          }
     else
     {
          cout<<a[i]<<" "<<b[j]<<" ";
          i+=1;
```

```
        j+=1;
        k+=2;
    }
    }
}
void selection (float a[],int n)
{
    int min,minind,temp;
  for(int i=0;i<n;i+=1)
  {   min=a[i];
     minind=i;
     for(int j=i+1;j<n;j+=1)
    {
        if(a[j]<min)
        {min=a[j];
          minind=j;}
    }

    if(minind!=i)
    {
        temp=a[i];
        a[i]=a[minind];
        a[minind]=temp;
    }
    }
```

```
}
int main()
{
   cout<<"\nMerging Two Sorted Arrays...\n";
     int n,m;
   cout<<"Enter the size of array 1:";
   cin>>n;
     float a[100000];
   cout<<"\nEnter the elements of array 1...\n";
   for(int i=0;i<n;i+=1)
     {
      cout<<"Element "<<i+1<<":";
      cin>>a[i];
     }
   cout<<"Enter the size of array 2:";
   cin>>m;
     float b[100000];
   cout<<"\nEnter the elements of array 2...\n";
   for(int i=0;i<m;i+=1)
     {
      cout<<"Element "<<i+1<<":";
      cin>>b[i];
     }
     selection(a,n);
     selection(b,m);
```

```
    cout<<"\nThe sorted arrays are:";
    cout<<"\nArray 1: ";
      For(i,n)
       cout<<a[i]<<" ";
    cout<<"\nArray 2: ";
      For(i,m)
       cout<<b[i]<<" ";
      merged(a,n,b,m);
      return 0;
}
```

- **OUTPUT**



```
C:\Users\harsh\Documents\data5.exe

Merging two sorted arrays...
Enter the size of array 1:7

Enter the elements of array 1...
Element 1:4
Element 2:2
Element 3:9
Element 4:0
Element 5:2
Element 6:1
Element 7:-8
Enter the size of array 2:4

Enter the elements of array 2...
Element 1:3
Element 2:6
Element 3:1
Element 4:2

The sorted arrays are:
Array 1: -8 0 1 2 2 4 9
Array 2: 1 2 3 6
Merged arrays are:-8 0 1 1 2 2 2 3 4 6 9
--------------------------------
Process exited after 15.33 seconds with return value 0
Press any key to continue . . .
```

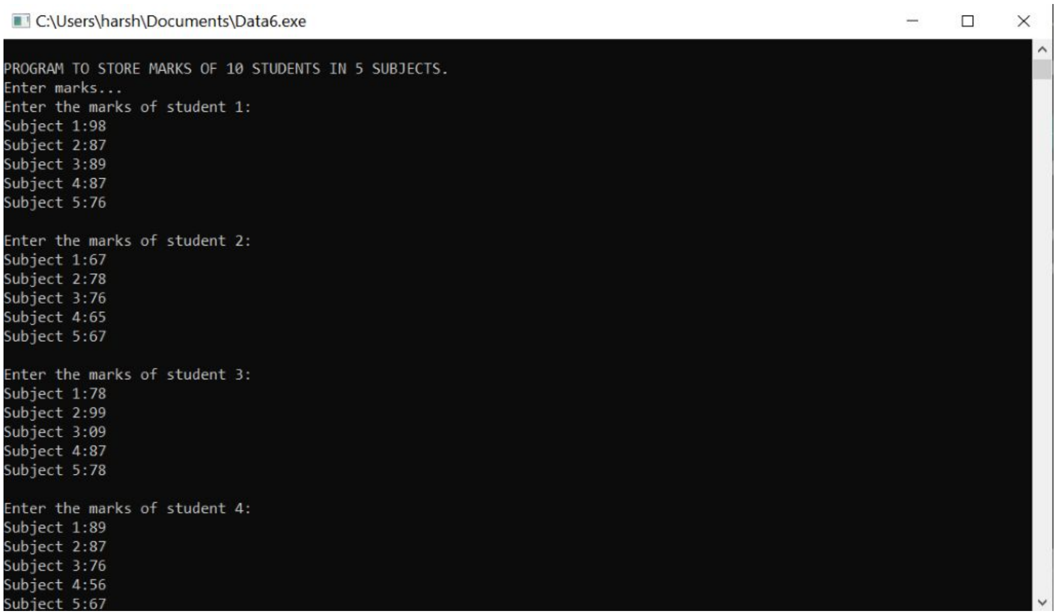
6. WAP to store marks obtained by 10 students in 5 courses in a two-dimensional array

- **C++ CODE**

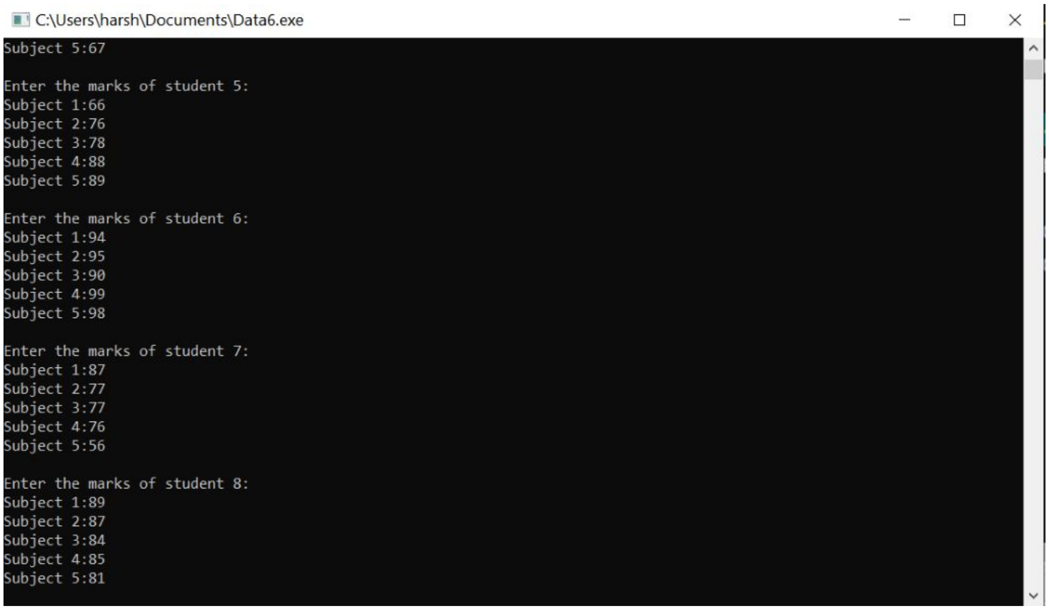
```
#include<bits/stdc++.h>
using namespace std;
int main()
{  cout<<"\nPROGRAM TO STORE MARKS OF 10 STUDENTS IN 5 SUBJECTS.";
       int marks[10][5];
    cout<<"\nEnter marks...";
    for(int i=0;i<10;i++)
       {
        cout<<"\nEnter the marks of student "<<i+1<<":\n";
        for(int j=0;j<5;j++)
       {
            cout<<"Subject "<<j+1<<":";
            cin>>marks[i][j];
       }
       }
    cout<<"\nThe marks are:";
        for(int i=0;i<10;i++)
       {
        cout<<"\nStudent "<<i+1<<":\n";
        for(int j=0;j<5;j++)
       {
            cout<<"Subject "<<j+1<<":";
            cout<<marks[i][j];
       }
```

```
        }
        return 0;
}
```

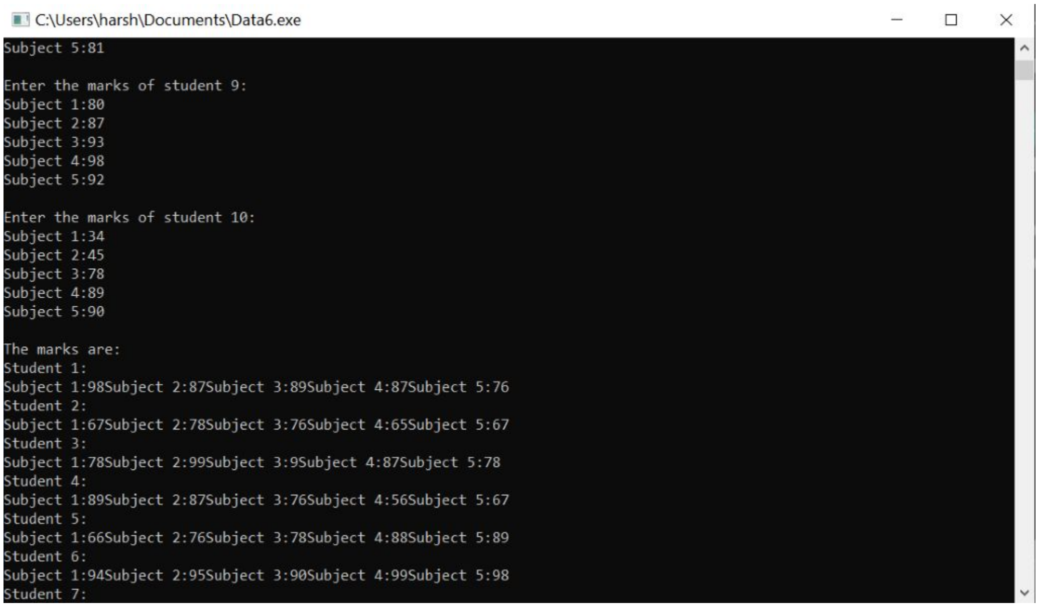
- **OUTPUT**



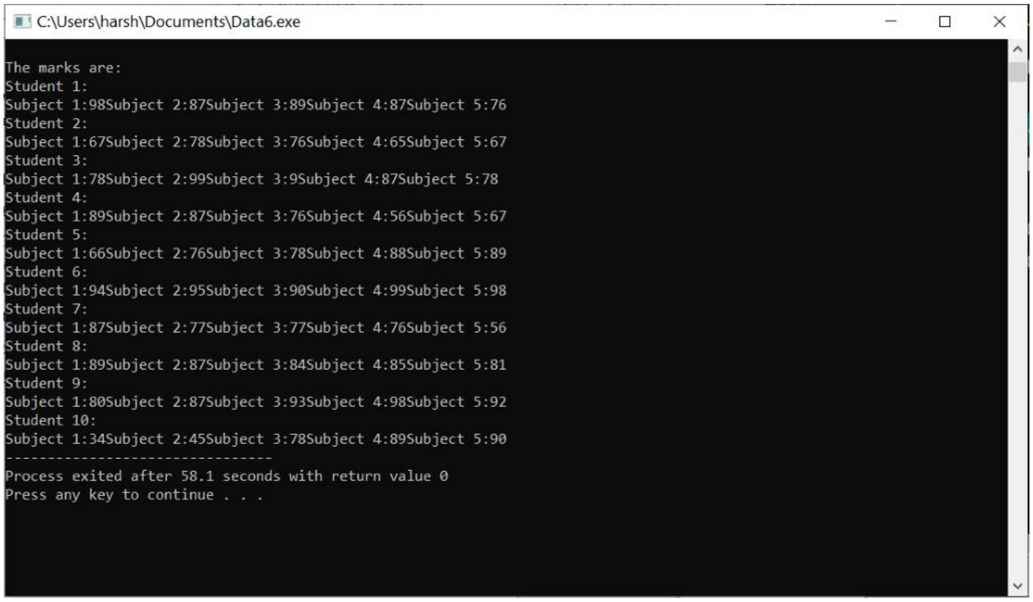
```
PROGRAM TO STORE MARKS OF 10 STUDENTS IN 5 SUBJECTS.
Enter marks...
Enter the marks of student 1:
Subject 1:98
Subject 2:87
Subject 3:89
Subject 4:87
Subject 5:76

Enter the marks of student 2:
Subject 1:67
Subject 2:78
Subject 3:76
Subject 4:65
Subject 5:67

Enter the marks of student 3:
Subject 1:78
Subject 2:99
Subject 3:09
Subject 4:87
Subject 5:78

Enter the marks of student 4:
Subject 1:89
Subject 2:87
Subject 3:76
Subject 4:56
Subject 5:67
```



```
Subject 5:67

Enter the marks of student 5:
Subject 1:66
Subject 2:76
Subject 3:78
Subject 4:88
Subject 5:89

Enter the marks of student 6:
Subject 1:94
Subject 2:95
Subject 3:90
Subject 4:99
Subject 5:98

Enter the marks of student 7:
Subject 1:87
Subject 2:77
Subject 3:77
Subject 4:76
Subject 5:56

Enter the marks of student 8:
Subject 1:89
Subject 2:87
Subject 3:84
Subject 4:85
Subject 5:81
```



```
Subject 5:81

Enter the marks of student 9:
Subject 1:80
Subject 2:87
Subject 3:93
Subject 4:98
Subject 5:92

Enter the marks of student 10:
Subject 1:34
Subject 2:45
Subject 3:78
Subject 4:89
Subject 5:90

The marks are:
Student 1:
Subject 1:98Subject 2:87Subject 3:89Subject 4:87Subject 5:76
Student 2:
Subject 1:67Subject 2:78Subject 3:76Subject 4:65Subject 5:67
Student 3:
Subject 1:78Subject 2:99Subject 3:9Subject 4:87Subject 5:78
Student 4:
Subject 1:89Subject 2:87Subject 3:76Subject 4:56Subject 5:67
Student 5:
Subject 1:66Subject 2:76Subject 3:78Subject 4:88Subject 5:89
Student 6:
Subject 1:94Subject 2:95Subject 3:90Subject 4:99Subject 5:98
Student 7:
```



```

The marks are:
Student 1:
Subject 1:98Subject 2:87Subject 3:89Subject 4:87Subject 5:76
Student 2:
Subject 1:67Subject 2:78Subject 3:76Subject 4:65Subject 5:67
Student 3:
Subject 1:78Subject 2:99Subject 3:9Subject 4:87Subject 5:78
Student 4:
Subject 1:89Subject 2:87Subject 3:76Subject 4:56Subject 5:67
Student 5:
Subject 1:66Subject 2:76Subject 3:78Subject 4:88Subject 5:89
Student 6:
Subject 1:94Subject 2:95Subject 3:90Subject 4:99Subject 5:98
Student 7:
Subject 1:87Subject 2:77Subject 3:77Subject 4:76Subject 5:56
Student 8:
Subject 1:89Subject 2:87Subject 3:84Subject 4:85Subject 5:81
Student 9:
Subject 1:80Subject 2:87Subject 3:93Subject 4:98Subject 5:92
Student 10:
Subject 1:34Subject 2:45Subject 3:78Subject 4:89Subject 5:90
--------------------------------
Process exited after 58.1 seconds with return value 0
Press any key to continue . . .
```

7. WAP to implement a linked list

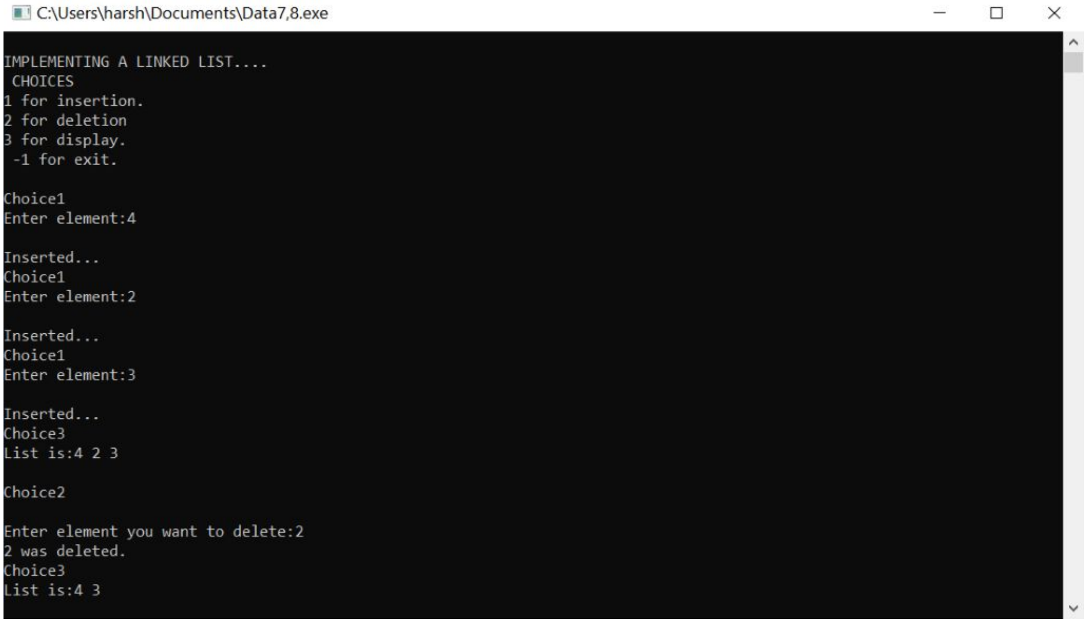
```
#include<iostream>
using namespace std;
struct node
{
        int data;
        struct node* next;
};
struct node* head=NULL;
struct node* last=NULL;
class links
{
        public:
        void insert (int n)
           {    struct node* temp;
                temp=new node;
                temp->data=n;
                //inserting at the end
                temp->next=NULL;
                if(last==NULL)
                   { last=temp;
                        head=last;
                }
                else
                {
                last->next=temp;
                last=temp;
                }
        }
                //deleting last element.
        int del(int ele)
```

```
{
  if(last==NULL )
    return -1;
else
{
    struct node* temp;
    struct node* tempb;
    temp=head;
    tempb=head;
    while(temp!=last && temp->data!=ele)
    {tempb=temp;
    temp=temp->next;
    }
    if(temp->data!=ele)
    return -1;
    else
    {
    if(temp==tempb)
        {
        head=temp->next;
        if(head==NULL)
            last==NULL;
        return (ele);
        }
    else
    {
        tempb->next=temp->next;
        If(tempb->next==NULL)
            last=tempb;
        temp=NULL;
        return (ele);
    }
```

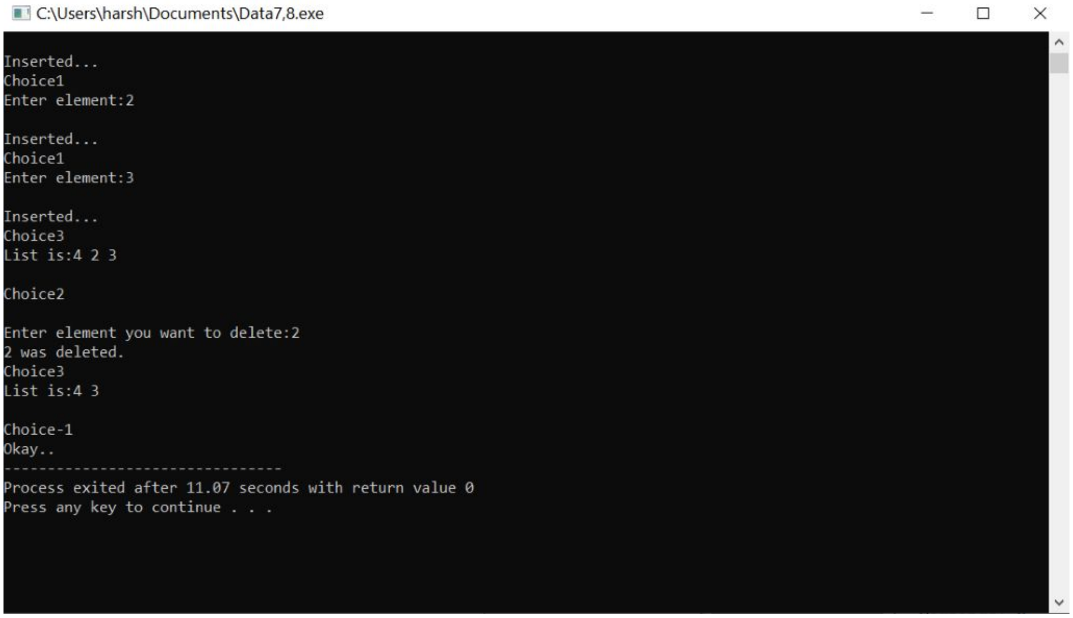
```
            }
        }
        }
        void disp()
        {
            if(last==NULL)
               cout<<"Empty list.";
            else
            {
            struct node* temp;
            temp=head;
            while(temp!=NULL)
            {
                cout<<temp->data<<":";
                temp=temp->next;
            }
               cout<<endl;
            }
        }

};
int main()
{
        links link;
    cout<<"\nIMPLEMENTING A LINKED LIST....\n";
    cout<<" CHOICES\n1 for insertion.\n2 for deletion \n3 for
display.\n -1 for exit.\n";
        int n;

        while(true)
        {cout<<"\nChoice";
    cin>>n;
    if(n!=-1)
    {
    switch(n)
    {
    case 1: int a;
            cout<<"Enter element:";
            cin>>a;
            link.insert(a);
            cout<<"\nInserted...";
        break;
    case 2: int r;
            cout<<"\nEnter element you want to delete:";
            cin>>r;
        int b;
        b = link.del(r);
        if(b!=-1)
            cout<<r<<" was deleted.";
        else
            cout<<"\nElement not found.";
        break;
    case 3:cout<<"List is:";
            link.disp();
        break;
    default: break;

    }
    }
    else
    { cout<<"Okay..";
    break;
    }
}
```

```
return 0;
}
```

- **OUTPUT**



```
IMPLEMENTING A LINKED LIST....
 CHOICES
1 for insertion.
2 for deletion
3 for display.
 -1 for exit.

Choice1
Enter element:4

Inserted...
Choice1
Enter element:2

Inserted...
Choice1
Enter element:3

Inserted...
Choice3
List is:4 2 3

Choice2

Enter element you want to delete:2
2 was deleted.
Choice3
List is:4 3
```



```
Inserted...
Choice1
Enter element:2

Inserted...
Choice1
Enter element:3

Inserted...
Choice3
List is:4 2 3

Choice2

Enter element you want to delete:2
2 was deleted.
Choice3
List is:4 3

Choice-1
Okay..
--------------------------------
Process exited after 11.07 seconds with return value 0
Press any key to continue . . .
```

8. WAP to insert and delete in a linked list

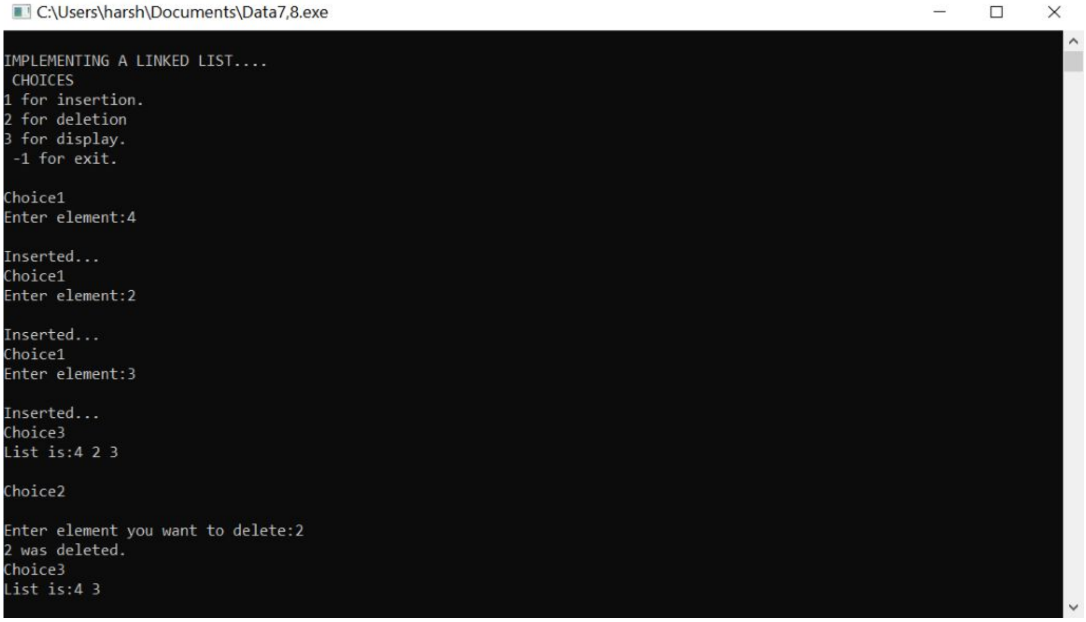
```
#include<iostream>
using namespace std;
struct node
{
        int data;
        struct node* next;
};
struct node* head=NULL;
struct node* last=NULL;
class links
{
        public:
        void insert (int n)
            {    struct node* temp;
                temp=new node;
                temp->data=n;
                //inserting at the end
                temp->next=NULL;
                if(last==NULL)
                    { last=temp;
                        head=last;
                }
                else
                {
                last->next=temp;
                last=temp;
                }
        }
        //deleting last element
        int del(int ele)
        {
```

```
  if(last==NULL )
      return -1;
else
{
      struct node* temp;
      struct node* tempb;
      temp=head;
      tempb=head;
      while(temp!=last && temp->data!=ele)
      {tempb=temp;
      temp=temp->next;
      }
      if(temp->data!=ele)
      return -1;
      else
      {
      if(temp==tempb)
            {
            head=temp->next;
            if(head==NULL)
                  last==NULL;
            return (ele);
            }
      else
      {
            tempb->next=temp->next;
            if(temp->next==NULL)
            last=tempb;

            return (ele);
      }
      }
```

```
        }
        }
        void disp()
        {
                if(last==NULL)
                   cout<<"Empty list.";
                else
                {
                struct node* temp;
                temp=head;
                while(temp!=NULL)
                {
                        cout<<temp->data<<" ";
                        temp=temp->next;
                }
                   cout<<endl;
                }
        }

};
int main()
{
        links link;
    cout<<"\n INSERTING AND DELETING FROM A LINKED
LIST....\n";
    cout<<" CHOICES\n1 for insertion.\n2 for deletion \n3 for
display.\n -1 for exit.\n";
        int n;

        while(true)
        {cout<<"\nChoice";
    cin>>n;
    if(n!=-1)
    {
    switch(n)
    {
    case 1: int a;
              cout<<"Enter element:";
              cin>>a;
              link.insert(a);
              cout<<"\nInserted...";
          break;
    case 2: int r;
              cout<<"\nEnter element you want to delete:";
              cin>>r;
          int b;
          b = link.del(r);
          if(b!=-1)
                cout<<r<<" was deleted.";
          else
                cout<<"\nElement not found.";
          break;
    case 3:cout<<"List is:";
              link.disp();
          break;
    default: break;

    }
    }
    else
    { cout<<"Okay..";
    break;
    }
}
```

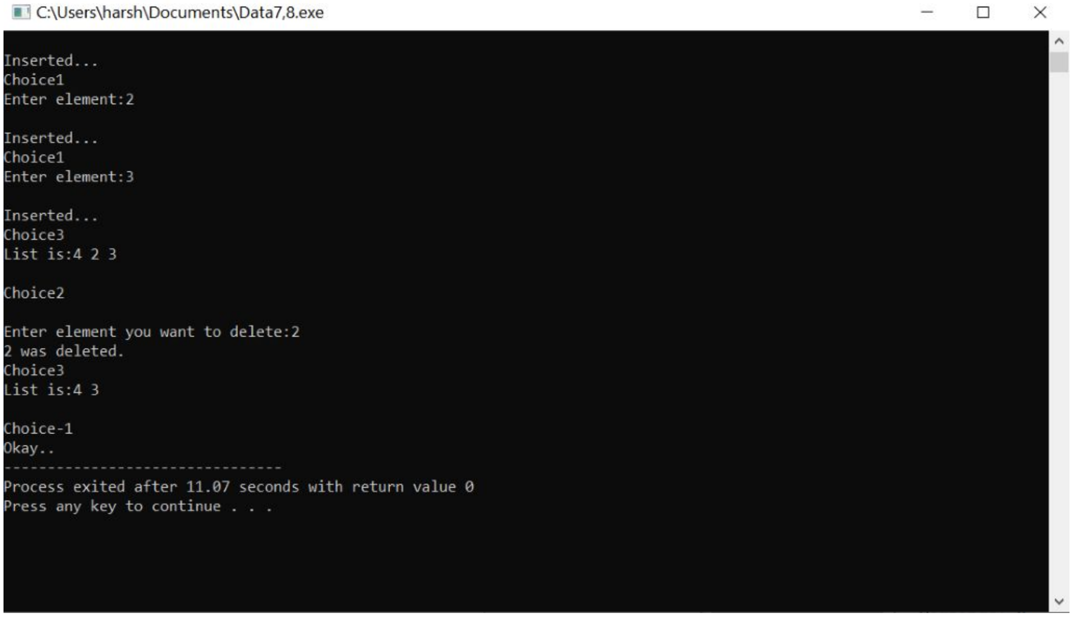
```
return 0;
}
```

- **OUTPUT**



C:\Users\harsh\Documents\Data7,8.exe

```
IMPLEMENTING A LINKED LIST....
 CHOICES
1 for insertion.
2 for deletion
3 for display.
 -1 for exit.

Choice1
Enter element:4

Inserted...
Choice1
Enter element:2

Inserted...
Choice1
Enter element:3

Inserted...
Choice3
List is:4 2 3

Choice2

Enter element you want to delete:2
2 was deleted.
Choice3
List is:4 3
```



C:\Users\harsh\Documents\Data7,8.exe

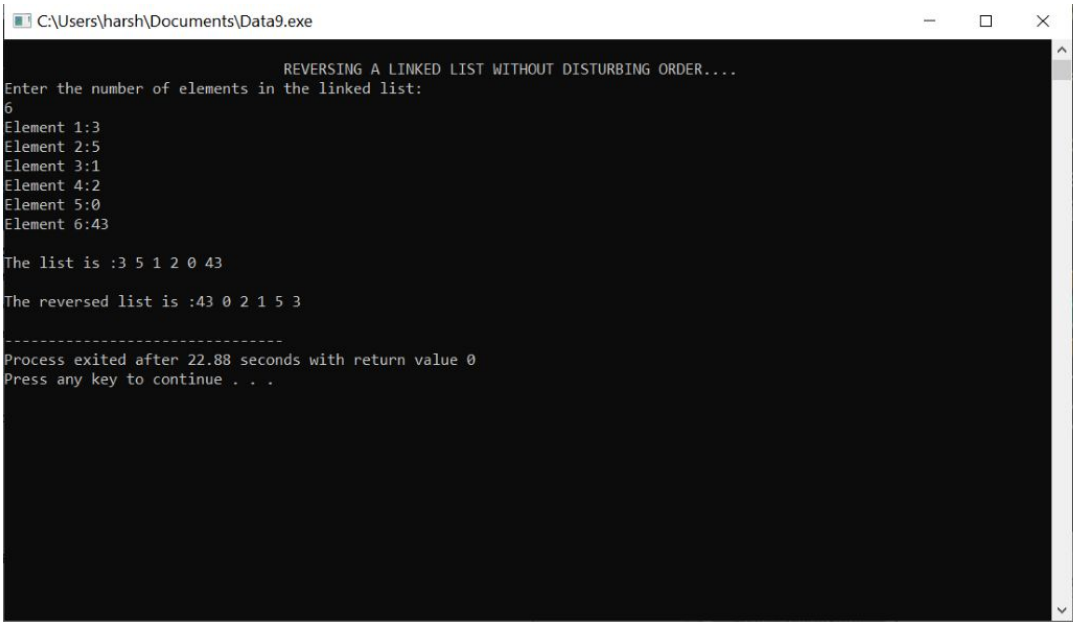
```
Inserted...
Choice1
Enter element:2

Inserted...
Choice1
Enter element:3

Inserted...
Choice3
List is:4 2 3

Choice2

Enter element you want to delete:2
2 was deleted.
Choice3
List is:4 3

Choice-1
Okay..
--------------------------------
Process exited after 11.07 seconds with return value 0
Press any key to continue . . .
```

9. WAP to print the elements of a linked list in reverse order without disturbing the linked list

```
#include<iostream>
using namespace std;
struct node
{
        int data;
        struct node* next;
};
struct node* head=NULL;
struct node* last=NULL;
class links
{
        public:
        void insert (int n)
            {    struct node* temp;
                temp=new node;
                temp->data=n;
                //inserting at the end
                temp->next=NULL;
                if(last==NULL)
                    { last=temp;
                        head=last;
                }
                else
                {
                last->next=temp;
                last=temp;
                }
        }
        void disp()
        {
```

```
        if(last==NULL)
            cout<<"Empty list.";
        else
        {
        struct node* temp;
        temp=head;
        while(temp!=NULL)
        {
                cout<<temp->data<<" ";
                temp=temp->next;
        }
            cout<<endl;
        }
}
void reverse(struct node* h)
{
        if(last==NULL)
        {cout<<"No element.\nList as it is...";
        }
        else
        {
        struct node* prev;
        struct node* curr;
        struct node* nex;
            prev=NULL;
            curr=head;
            nex=NULL;
        while(curr!=NULL)
            {    nex=curr->next;
                curr->next=prev;
                prev=curr;
                curr=nex;
                }
                head=prev;
                }
        }
};
int main()
{
        links link;
    cout<<"\n\t\t\t\tREVERSING A LINKED LIST WITHOUT
DISTURBING ORDER....\n";
    cout<<"Enter the number of elements in the linked list:\n";
        int n,ele;
    cin>>n;
    for(int i=0;i<n;i++)
        {
        cout<<"Element "<<i+1<<":";
        cin>>ele;
        link.insert(ele);
        }
    cout<<"\nThe list is :";
    link.disp();
    cout<<"\nThe reversed list is :";
    link.reverse(head);
    link.disp();
return 0;
}
```

- **OUTPUT**



```
                                REVERSING A LINKED LIST WITHOUT DISTURBING ORDER....
Enter the number of elements in the linked list:
6
Element 1:3
Element 2:5
Element 3:1
Element 4:2
Element 5:0
Element 6:43

The list is :3 5 1 2 0 43

The reversed list is :43 0 2 1 5 3

--------------------------------
Process exited after 22.88 seconds with return value 0
Press any key to continue . . .
```

10. WAP to reverse a linked list

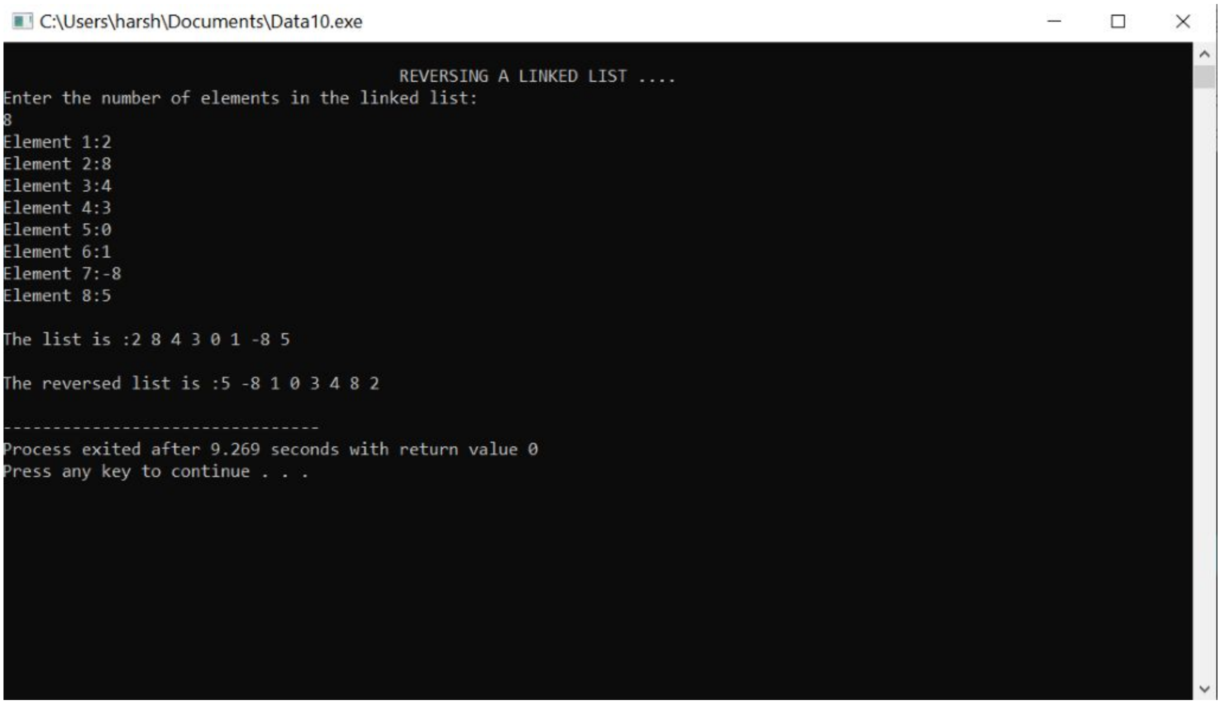
```
#include<iostream>
using namespace std;
void swap(int* a,int* b)
{
        int temp;
        temp= *a;
        *a = *b;
        *b = temp;

}
struct node
{
        int data;
        struct node* next;
};
struct node* head=NULL;
struct node* last=NULL;
class links
{
        public:
                void insert (int n)
              {   struct node* temp;
                temp=new node;
                temp->data=n;
                //inserting at the end
                temp->next=NULL;
                if(last==NULL)
```

```
   { last=temp;
    head=last;
    }
else
{
    last->next=temp;
    last=temp;
}
}
void disp()
{
if(last==NULL)
   cout<<"Empty list.";
else
{
    struct node* temp;
    temp=head;
    while(temp!=NULL)
    {
      cout<<temp->data<<" ";
    temp=temp->next;
    }
   cout<<endl;
}
}
void reverseSq(struct node* h)
{
if(h==NULL)
```

```
           cout<<"No list to reverse.";
        else
        {
            node* tempb;
            node* tempf;
            node* temp;
           tempb=head;
            temp=head;
            int size=1;
            while(temp->next!=NULL)
            {temp=temp->next;
            size++;}
            int i,j;
           i=0;
            while(i<size/2)
           {   j=0;
             tempf=head;
            while(j<size-i-1)
                {tempf=tempf->next;
               j++;}
            swap( &tempb->data, &tempf->data);
             tempb=tempb->next;
             i++;                   }
        }
        }
};
int main()
{     links link;
```

```
   cout<<"\n\t\t\t\t\tREVERSING A LINKED LIST ....\n";
   cout<<"Enter the number of elements in the linked
list:\n";
       int n,ele;
   cin>>n;
   for(int i=0;i<n;i++)
       {
        cout<<"Element "<<i+1<<":";
        cin>>ele;
        link.insert(ele);
       }     cout<<"\nThe list is :";
   link.disp();
   cout<<"\nThe reversed list is :";
   link.reverseSq(head);
   link.disp();
return 0;}
```

- **OUTPUT**



```
                                        REVERSING A LINKED LIST ....
Enter the number of elements in the linked list:
8
Element 1:2
Element 2:8
Element 3:4
Element 4:3
Element 5:0
Element 6:1
Element 7:-8
Element 8:5

The list is :2 8 4 3 0 1 -8 5

The reversed list is :5 -8 1 0 3 4 8 2

--------------------------------
Process exited after 9.269 seconds with return value 0
Press any key to continue . . .
```

11.WAP to add two polynomials using a linked list

- **C++ CODE**

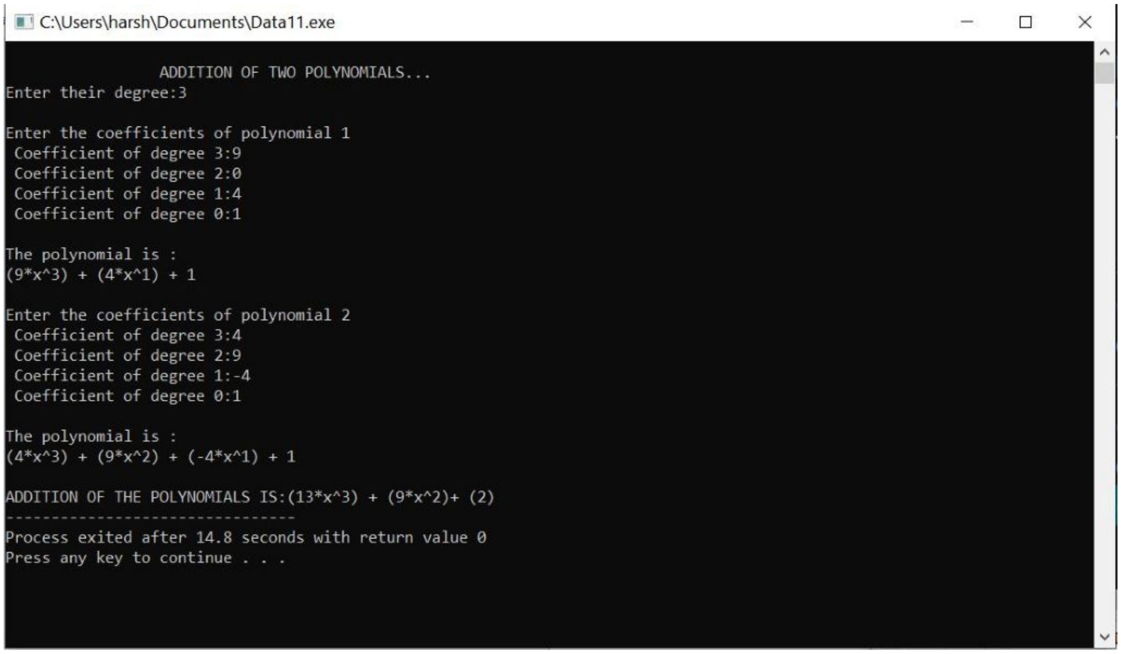
```
#include<bits/stdc++.h>
using namespace std;
struct node
{
      int data;
      node *next;
};
class links
{
 public: struct node* head=NULL;
             struct node* tail=NULL;
             void insert (int n)
           {   struct node* temp;
             temp=new node;
             temp->data=n;
             //inserting at the end
             temp->next=NULL;
             if(tail==NULL)
                  { tail=temp;
                    head=tail;
                    }
             else
             {
                    tail->next=temp;
                    tail=temp;
```

```
}
}
void show(int degree)
{
 cout<<"\nThe polynomial is :\n";
int i=degree;
node *temp;
temp=head;
while(temp!=NULL)
{
        if(temp->data==0)
        {
          i--;
        temp=temp->next;
        continue;}
        else
        {
        if(temp!=tail)
        { cout<<"("<<temp->data<<"*x^"<<i<<") + ";
        }
        else
        {
            cout<<temp->data;

        }
          i--;
        temp=temp->next;
        }
```

```
            }
             cout<<endl;
            }
            int coef(int step)
            {
            node *temp=head;
            int i=0;
            while(i<step)
                   {temp=temp->next;
                 i++;}
            return(temp->data);
            }
};
int main()
{    links p1,p2;
    cout<<"\n\t\t ADDITION OF TWO POLYNOMIALS...";
    cout<<"\nEnter their degree:";
      int deg,c;
    cin>>deg;
    cout<<"\nEnter the coefficients of polynomial 1\n";
    for(int i=0;i<=deg;i++)
      {cout<<" Coefficient of degree "<<deg-i<<":";
    cin>>c;
      p1.insert(c);
      }
      p1.show(deg);
    cout<<"\nEnter the coefficients of polynomial 2\n";
    for(int i=0;i<=deg;i++)
```

```
        {cout<<" Coefficient of degree "<<deg-i<<":";
    cin>>c;
        p2.insert(c);
        }
        p2.show(deg);
        int res;
    cout<<"\nADDITION OF THE POLYNOMIALS IS:";
    for(int i=0;i<=deg;i++)
        {
        res=p1.coef(i)+p2.coef(i);
        if(res==0)
        {continue;
        }
        else
         {    if(i==deg)
              cout<<"+ ("<<res<<")";
                 else
              { if(i!=0)
              cout<<"+ ("<<res<<"*x^"<<deg-i<<")";
                 else
              cout<<"("<<res<<"*x^"<<deg-i<<") ";
        }
        }
        }
        return 0;
}
```

- **OUTPUT**



```
                 ADDITION OF TWO POLYNOMIALS...
Enter their degree:3

Enter the coefficients of polynomial 1
 Coefficient of degree 3:9
 Coefficient of degree 2:0
 Coefficient of degree 1:4
 Coefficient of degree 0:1

The polynomial is :
(9*x^3) + (4*x^1) + 1

Enter the coefficients of polynomial 2
 Coefficient of degree 3:4
 Coefficient of degree 2:9
 Coefficient of degree 1:-4
 Coefficient of degree 0:1

The polynomial is :
(4*x^3) + (9*x^2) + (-4*x^1) + 1

ADDITION OF THE POLYNOMIALS IS:(13*x^3) + (9*x^2)+ (2)
--------------------------------
Process exited after 14.8 seconds with return value 0
Press any key to continue . . .
```

12. WAP to implement a doubly linked list

- **C++ CODE**

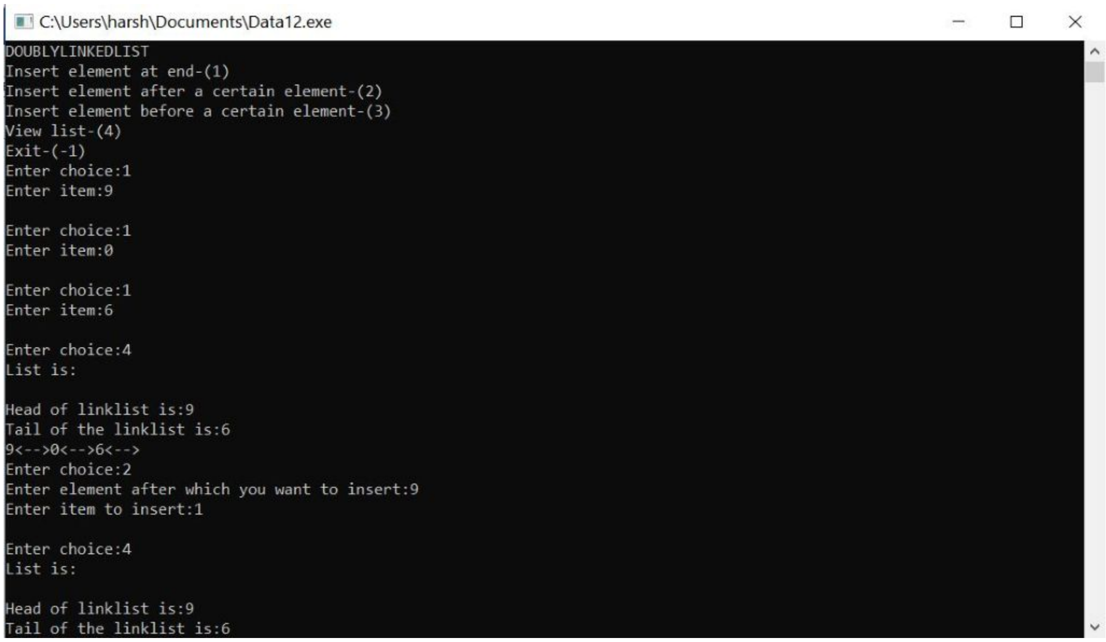
```
#include<iostream>
using namespace std;
struct node
{
    int data;
    struct node* prev;
    struct node* next;
};
typedef struct node nd;
nd* head;
nd* tail;
class links
{
    public://head=NULL;
    //tail=NULL;
    void insertN(int d)
    { nd* temp;
        temp = new node;
        temp->data= d;
        temp->next=NULL;
        temp->prev=NULL;

        if(head==NULL)
            head=tail=temp;
        else
        {
                tail->next=temp;
                temp->prev=tail;
                tail=temp;
}
}
void insertAft(int d,int s)
    {    nd* temp;
             nd* trv;
                if(head==NULL)
             cout<<"\nEmpty list...";
        else
        {
               trv=head;
                while(trv->data!=s && trv->next!=NULL)
                    trv=trv->next;
                if(trv->data!=s)
                    cout<<"\nElement not found.";
                else
                {
                temp= new node;
                temp->data=d;
                temp->next=trv->next;
                    trv->next=temp;
                if(trv->next!=NULL)
                        trv->next->prev=temp;
                else
                        tail=temp;
                 temp->prev=trv;
        }
    }
}
void insertBef(int d, int s)
{     nd* temp;
    nd* trv;
    if(head==NULL)
      cout<<"Empty list...";
    else
    {
      trv=head;
        while(trv->next!=NULL && trv->data!=s)
         trv=trv->next;
        if(trv->data!=s)
         cout<<"\nElement not found...";
        else
        {
        temp=new node;
        temp->data=d;
        temp->next=trv;
        temp->prev=trv->prev;
        if(trv!=head)
           trv->prev->next=temp;
        else
            head=temp;
       trv->prev=temp;
        }
    }
```

```
      }
      void show()
      {          nd* temp;
             temp=head;
             if(head==NULL)
                 cout<<"\nList is empty...";
             else
              {   cout<<"\n";
                 cout<<"\nHead of linklist is:"<<head->data;
                 cout<<"\nTail of the linklist
is:"<<tail->data<<"\n";
                   while(temp!=NULL)
                   {
                 cout<<temp->data<<"<-->";
                   temp=temp->next;}
             }
      }
};
int main()
{
      links ob;
   cout<<"DOUBLYLINKEDLIST\nInsert element at
end-(1)\nInsert element after a certain element-(2)\nInsert
element before a certain element-(3)\nView
list-(4)\nExit-(-1)";
      int choice,it,ele;
      while (true)
   {   cout<<"\nEnter choice:";
```

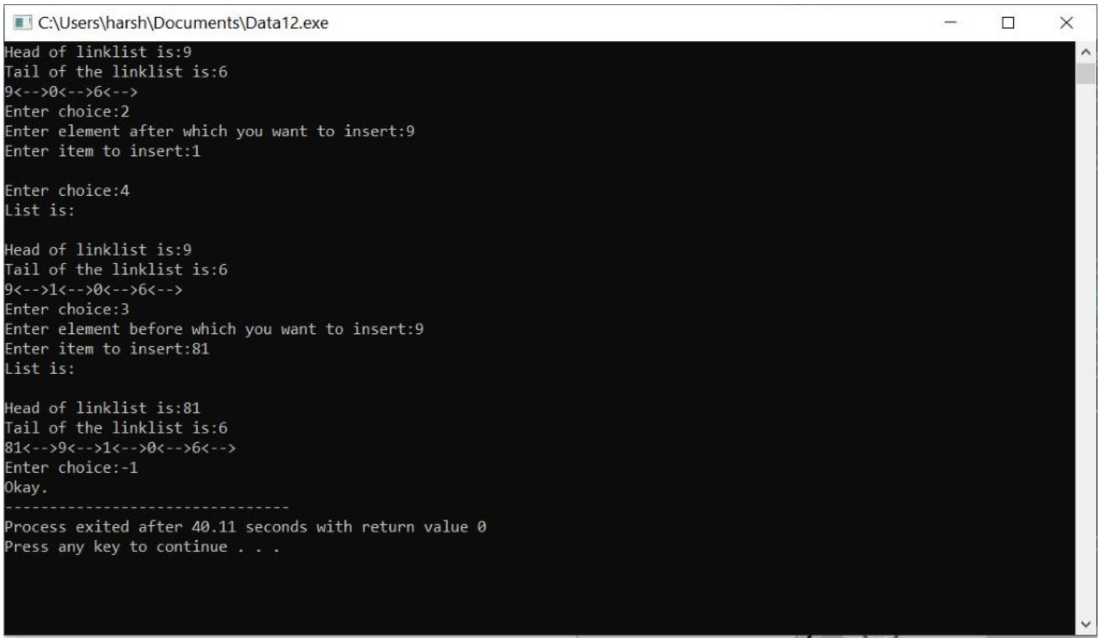
```
     cin>>choice;
    if(choice!=-1)
    {           switch(choice)
        {
        case 1: cout<<"Enter item:";
             cin>>it;
             ob.insertN(it);
            break;
        case 2: cout<<"Enter element after which you
want to insert:";
             cin>>ele;
             cout<<"Enter item to insert:";
             cin>>it;
             ob.insertAft(it,ele);
            break;
        case 3: cout<<"Enter element before which you
want to insert:";
             cin>>ele;
             cout<<"Enter item to insert:";
             cin>>it;
             ob.insertBef(it,ele);
        case 4: cout<<"List is:";
             ob.show();
            break;
        default: cout<<"invalid enter again.";
        }
    }
    else
```

```
{           cout<<"Okay.";
        break;
}
}
return 0; }
```

- **OUTPUT**



```
DOUBLYLINKEDLIST
Insert element at end-(1)
Insert element after a certain element-(2)
Insert element before a certain element-(3)
View list-(4)
Exit-(-1)
Enter choice:1
Enter item:9

Enter choice:1
Enter item:0

Enter choice:1
Enter item:6

Enter choice:4
List is:

Head of linklist is:9
Tail of the linklist is:6
9<-->0<-->6<-->
Enter choice:2
Enter element after which you want to insert:9
Enter item to insert:1

Enter choice:4
List is:

Head of linklist is:9
Tail of the linklist is:6
```



```
Head of linklist is:9
Tail of the linklist is:6
9<-->0<-->6<-->
Enter choice:2
Enter element after which you want to insert:9
Enter item to insert:1

Enter choice:4
List is:

Head of linklist is:9
Tail of the linklist is:6
9<-->1<-->0<-->6<-->
Enter choice:3
Enter element before which you want to insert:9
Enter item to insert:81
List is:

Head of linklist is:81
Tail of the linklist is:6
81<-->9<-->1<-->0<-->6<-->
Enter choice:-1
Okay.
--------------------------------
Process exited after 40.11 seconds with return value 0
Press any key to continue . . .
```

13. WAP to implement stack using an array

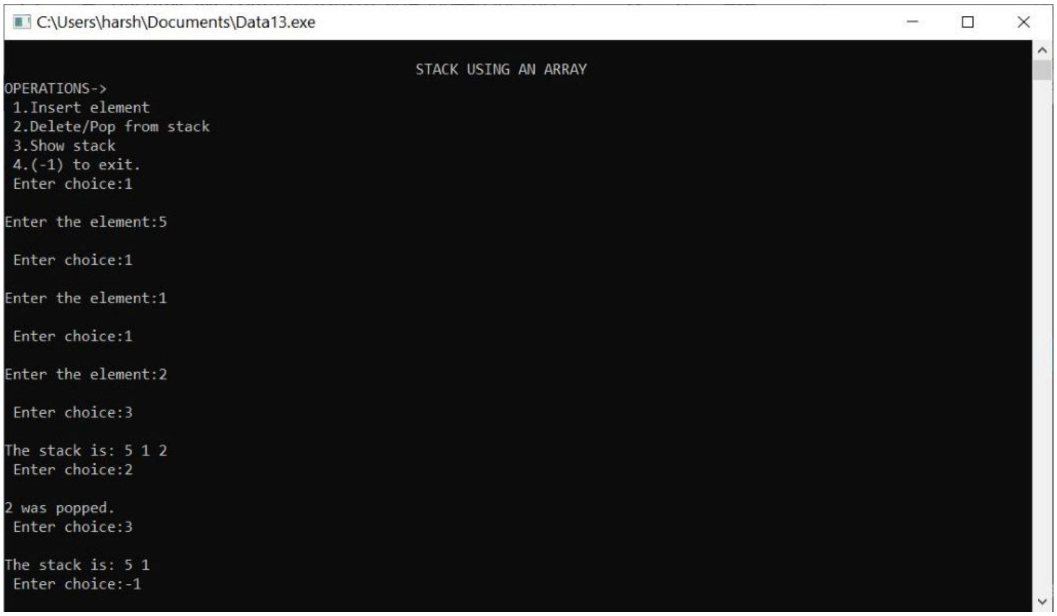
- **C++ CODE**

```
#include<bits/stdc++.h>
#define MAX 15
using namespace std;
int st[MAX];
int top=0;
void insert(int ele)
{
        if(top==MAX)
        {cout<<"\nst overflow.";
        return;}
        else
        {
         st[top++]=ele;
        }
}
void pop()
{
        if(top==0)
        {
         cout<<"\nStack underflow.";
        return;}
        else
        {
         cout<<endl<<st[--top]<<" was popped.";
        }
}
```

```
void show()
{
    if(top==0)    cout<<"\nStack underflow.";
    else
  {   cout<<"\nThe stack is: ";
     for(int i=0;i<top;i++)
        cout<<st[i]<<" ";
    }
}
int main()
{
  ios::sync_with_stdio(false);
    int choice;
  cout<<"\n\t\t\t\t\t\tSTACK USING AN ARRAY";
  cout<<"\nOPERATIONS->\n 1.Insert element\n
2.Delete/Pop from stack\n 3.Show stack\n 4.(-1) to exit.";
  while(1)
    {
     cout<<"\n Enter choice:";
     cin>>choice;
    if(choice==-1)      break;
    else
    {
        switch(choice)
        {
         case(1):{ int ele;
              cout<<"\nEnter the element:";
              cin>>ele;
```

```
                insert(ele);
                break;
            }
             case(2):{ pop();
                break;
            }
            case (3):{ show();
                break;
            }
             default:{
                break;
            }
                }
            }
        }
        return 0;
}
```

- **OUTPUT**



```
                                        STACK USING AN ARRAY
OPERATIONS->
 1.Insert element
 2.Delete/Pop from stack
 3.Show stack
 4.(-1) to exit.
 Enter choice:1

Enter the element:5

 Enter choice:1

Enter the element:1

 Enter choice:1

Enter the element:2

 Enter choice:3

The stack is: 5 1 2
 Enter choice:2

2 was popped.
 Enter choice:3

The stack is: 5 1
 Enter choice:-1
```

14. WAP to implement stack using linked list

- **C++ CODE**

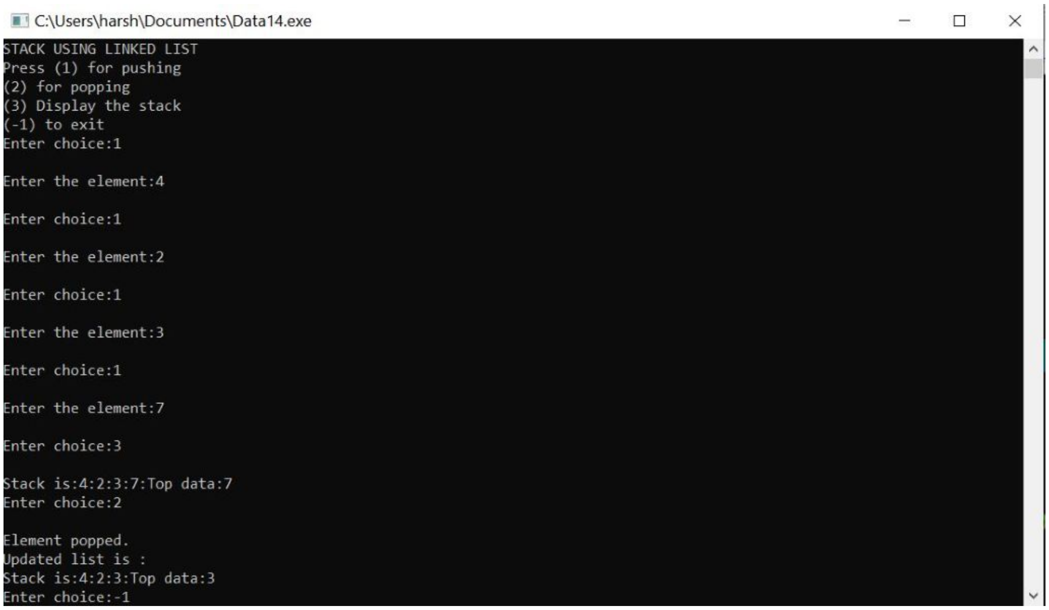
```
#include<iostream>
using namespace std;
struct node
{
      int data;
      struct node* next;
};
typedef struct node nd;
nd* top;
nd* head;
class st
{
      public://top=NULL;
            //head=NULL;
      void push(int item)
      {
          nd* temp;
            temp=new node;
            temp->data=item;
            temp->next=NULL;
            if(top==NULL)
            {top=temp;
            head=top;}
            else
            {top->next=temp;
```

```
        top=temp;}
}
void pop()
{
        if(top==NULL)
         cout<<"Stack underflow...";
        else
     {    int dat=top->data;
         nd* temp;
        temp=head;
        //important point...
        if(temp==top)
              {top=NULL;
              head=NULL;
              }
        else
         { while (temp->next!=top)
              temp=temp->next;
        top=temp;
        //important point...
        top->next=NULL;
        }
        }
}
void disp()
{
        if(top==NULL)
         cout<<"Empty stack.";
```

```
            else
           {    cout<<"\nStack is:";
             nd* temp;
            temp=head;
            while(temp!=NULL)
                 {cout<<temp->data<<":";
                 temp=temp->next;}
             cout<<"Top data:"<<top->data;

            }
       }
       };
int main()
{
   st s;
       int choice,ele;
   cout<<"\nSTACK USING LINKED LIST";
   cout<<"\nPress (1) for pushing\n(2) for popping\n(3)
Display the stack\n(-1) to exit";
       while(true)
       {
        cout<<"\nEnter choice:";
        cin>>choice;
       if(choice==-1) break;
       else
       {
            if(choice==1)
            {
```

```
         cout<<"\nEnter the element:";
         cin>>ele;
         s.push(ele);
        }
        if(choice==2)
        {
         s.pop();
         cout<<"\nElement popped.";
         cout<<"\nUpdated list is :";
         s.disp();
        }
        if(choice==3)
        {
         s.disp();
        }
    }
    }
    return 0;
}
```

- **OUTPUT**



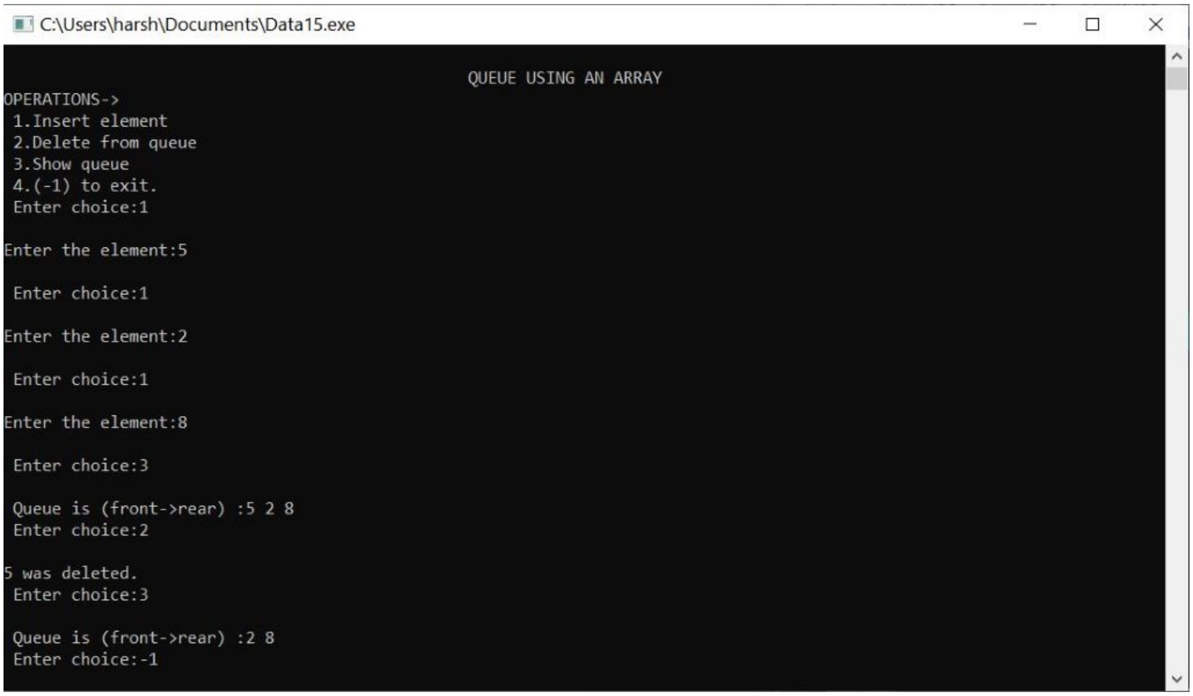
```
STACK USING LINKED LIST
Press (1) for pushing
(2) for popping
(3) Display the stack
(-1) to exit
Enter choice:1

Enter the element:4

Enter choice:1

Enter the element:2

Enter choice:1

Enter the element:3

Enter choice:1

Enter the element:7

Enter choice:3

Stack is:4:2:3:7:Top data:7
Enter choice:2

Element popped.
Updated list is :
Stack is:4:2:3:Top data:3
Enter choice:-1
```

15. WAP to implement queue using an array

```
#include<bits/stdc++.h>
#define MAX 10
using namespace std;
int Q[MAX];
int front=0;
int rear=0;
void insert(int ele)
{    if(rear!=MAX)
        Q[rear++]=ele;
        else cout<<"\nQueue is full.";
}
void del()
{
        if(front==rear)
        {
         cout<<"\nEmpty queue.";
        if(front!=0)
                front=rear=0;
        }

        else
        {
         cout<<endl<<Q[front++]<<" was deleted.";
        if(front==rear)
                front=rear=0;
        }
```

```
}
void show()
{
       if(front==rear)
        cout<<"\nEmpty queue.";
       else
       {  cout<<"\n Queue is (front->rear) :";
    for(int i=front;i<rear;i++)
            cout<<Q[i]<<" ";
       }
}
int main()
{
    ios::sync_with_stdio(false);
       int choice;
    cout<<"\n\t\t\t\t\t\tQUEUE USING AN ARRAY";
    cout<<"\nOPERATIONS->\n 1.Insert element\n 2.Delete from queue\n 3.Show queue\n 4.(-1) to exit.";
    while(1)
       {
        cout<<"\n Enter choice:";
        cin>>choice;
       if(choice==-1)      break;
       else
       {
              switch(choice)
              {
                case(1):{ int ele;
                cout<<"\nEnter the element:";
                cin>>ele;
               insert(ele);
               break;
          }
           case(2):{ del();
               break;
          }
          case (3):{ show();
               break;
          }
           default:{
               break;
          }
               }
          }
     }
     return 0;
}
```

- **OUTPUT**



```
                                                QUEUE USING AN ARRAY
OPERATIONS->
 1.Insert element
 2.Delete from queue
 3.Show queue
 4.(-1) to exit.
 Enter choice:1

Enter the element:5

 Enter choice:1

Enter the element:2

 Enter choice:1

Enter the element:8

 Enter choice:3

 Queue is (front->rear) :5 2 8
 Enter choice:2

5 was deleted.
 Enter choice:3

 Queue is (front->rear) :2 8
 Enter choice:-1
```

16. WAP to implement queue using linked list

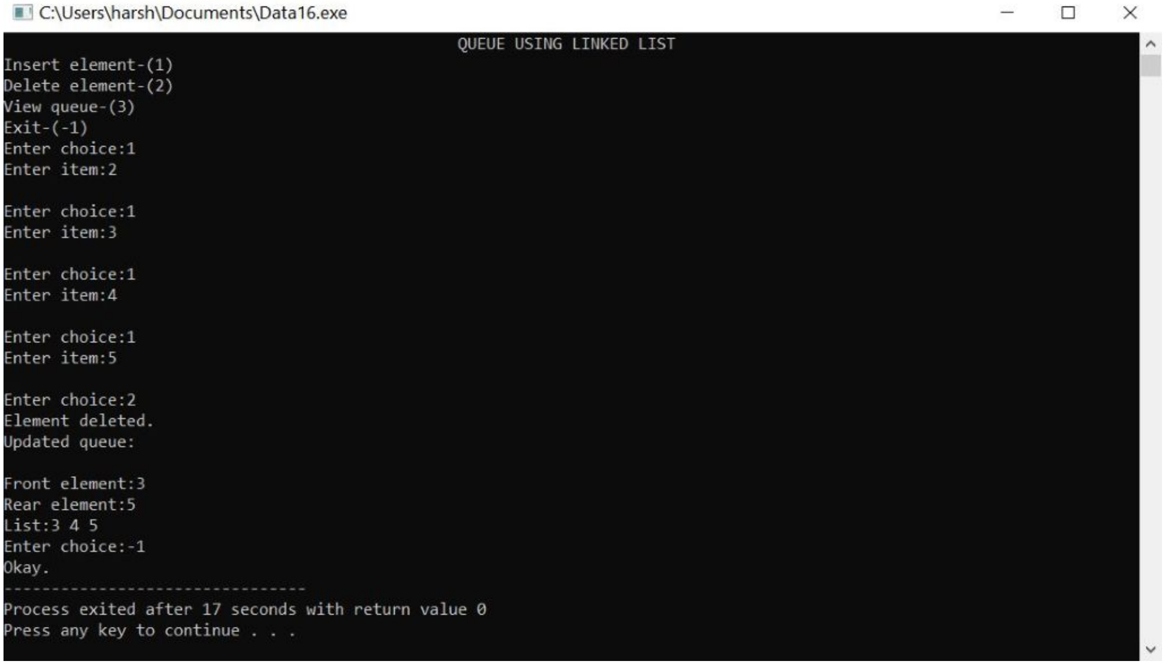
- **C++ CODE**

```
#include<iostream>
using namespace std;
struct node
{
    int data;
    struct node* next;
};
typedef struct node nd;
nd* rear;
nd* front;
class Q
{
    public: void insert(int item)
        {
          nd* temp;
        temp=new node;
        temp->data=item;
        temp->next=NULL;
        //important...if you check the condition that rear == front then it will give an error why?
        //as when you have inserted the first element , you still would have front==rear and not
        //just in the emty queue case...
        if(rear==NULL)
            {rear=temp;
        front=rear;}
else
{
        rear->next=temp;
        rear=temp;
}
}
int del()
{
if(front==NULL)
        return -1;
else
{
        int dat=front->data;
        if(front==rear)
        {front=NULL;
        rear=NULL;
        return dat;}
        else
        {
        front=front->next;
        return dat;
        }
}
}
void disp()
{
if(front==NULL)
```

```
           cout<<"\nEmpty queue...";
        else
         {   nd* temp;
             temp=front;
           cout<<"\nFront element:"<<front->data;
           cout<<"\nRear element:"<<rear->data;
           cout<<"\nList:";
             while(temp!=NULL)
             {cout<<temp->data<<" ";
             temp=temp->next;}
        }
        }
};
int main()
{
    Q q;
  cout<<"\n\t\t\t\t\t\tQUEUE USING LINKED LIST\nInsert
element-(1)\nDelete element-(2)\nView
queue-(3)\nExit-(-1)";
    int choice,it;
    while (true)
  {   cout<<"\nEnter choice:";
     cin>>choice;
    if(choice!=-1)
    {
        switch(choice)
        {
        case 1: cout<<"Enter item:";
```

```
             cin>>it;
             q.insert(it);
            break;
        case 2: int b;
            b = q.del();
            if(b!=-1)
                 {cout<<"Element deleted.\nUpdated
queue:\n";
                q.disp();}
            else
                cout<<"\nEmpty queue.";
            break;
        case 3: cout<<"QUEUE is:-";
             q.disp();
            break;
        default: cout<<"invalid enter again.";
        }
    }
    else
     {          cout<<"Okay.";
        break;
    }
  }   return 0;
}
```

- **OUTPUT**



```
                                                QUEUE USING LINKED LIST
Insert element-(1)
Delete element-(2)
View queue-(3)
Exit-(-1)
Enter choice:1
Enter item:2

Enter choice:1
Enter item:3

Enter choice:1
Enter item:4

Enter choice:1
Enter item:5

Enter choice:2
Element deleted.
Updated queue:

Front element:3
Rear element:5
List:3 4 5
Enter choice:-1
Okay.
--------------------------------
Process exited after 17 seconds with return value 0
Press any key to continue . . .
```

17. WAP to implement circular queue using array

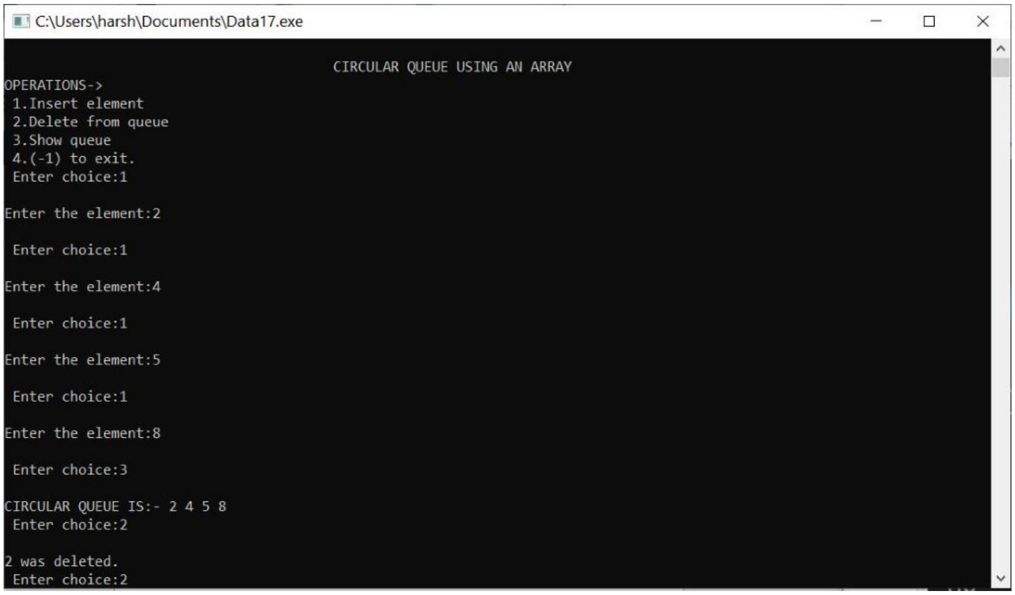
- **C++ CODE**

```
#include<iostream>
#define MAX 6
using namespace std;
int front=0;
int rear=0;
int arr[MAX];
void ins(int d)
{
        if(front==((rear+1)%MAX))
         cout<<"Queue full.";
        else
        {
         arr[rear]=d;
        rear=(rear+1)%MAX;
        }
}
int del()
{
        if(front==rear)
        return -1;
        else
        {
        int temp;
        temp=arr[front];
        front=(front+1)%MAX;
        return temp;
        }
}
void disp()
{
        if(front==rear)
        {
         cout<<"Empty queue.";
        }
        else
    {    cout<<"\nCIRCULAR QUEUE IS:- ";
        if(rear>front)
        {for(int i=front;i<rear;i++)
                { cout<<arr[i]<<" ";
                }
        }
        else
        {
         for(int i=front;i<MAX;i++)
        {
             cout<<arr[i]<<" ";
        }
         for(int i=0;i<rear;i++)
        {
             cout<<arr[i]<<" ";
        }
        }
} }
int main()
```

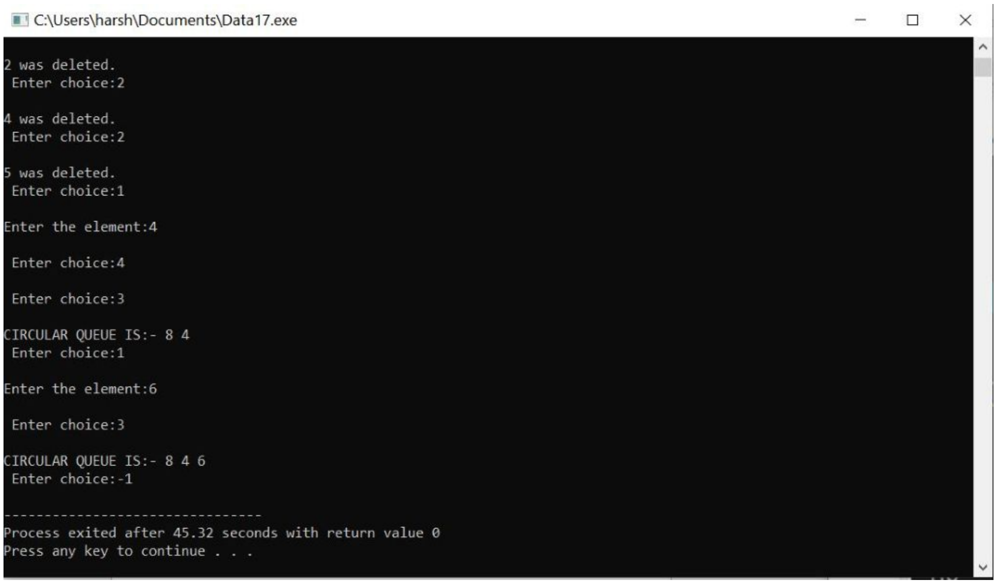
```
{    ios::sync_with_stdio(false);
     int choice;
   cout<<"\n\t\t\t\t\tCIRCULAR QUEUE USING AN
ARRAY";
   cout<<"\nOPERATIONS->\n 1.Insert element\n 2.Delete
from queue\n 3.Show queue\n 4.(-1) to exit.";
   while(1)
     {
      cout<<"\n Enter choice:";
      cin>>choice;
     if(choice==-1)      break;
     else
     { switch(choice)
          {
           case(1):{ int ele;
                 cout<<"\nEnter the element:";
                 cin>>ele;
               ins(ele);
               break;
          }
           case(2):{ int fr=del();
               if(fr==-1) cout<<"Empty queue.";
               else cout<<"\n"<<fr<<" was deleted.";
               break;
          }
          case (3):{ disp();
               break;
          }
```

```
                default:{
                        break;
              }
                        }
              }
        }
        return 0;
}
```

- **OUTPUT**



```
C:\Users\harsh\Documents\Data17.exe

                                 CIRCULAR QUEUE USING AN ARRAY
OPERATIONS->
 1.Insert element
 2.Delete from queue
 3.Show queue
 4.(-1) to exit.
 Enter choice:1

Enter the element:2

 Enter choice:1

Enter the element:4

 Enter choice:1

Enter the element:5

 Enter choice:1

Enter the element:8

 Enter choice:3

CIRCULAR QUEUE IS:- 2 4 5 8
 Enter choice:2

2 was deleted.
 Enter choice:2
```



```
2 was deleted.
 Enter choice:2

4 was deleted.
 Enter choice:2

5 was deleted.
 Enter choice:1

Enter the element:4

 Enter choice:4

 Enter choice:3

CIRCULAR QUEUE IS:- 8 4
 Enter choice:1

Enter the element:6

 Enter choice:3

CIRCULAR QUEUE IS:- 8 4 6
 Enter choice:-1

--------------------------------
Process exited after 45.32 seconds with return value 0
Press any key to continue . . .
```

18. WAP to implement a priority queue using a linked list

- **C++ CODE**

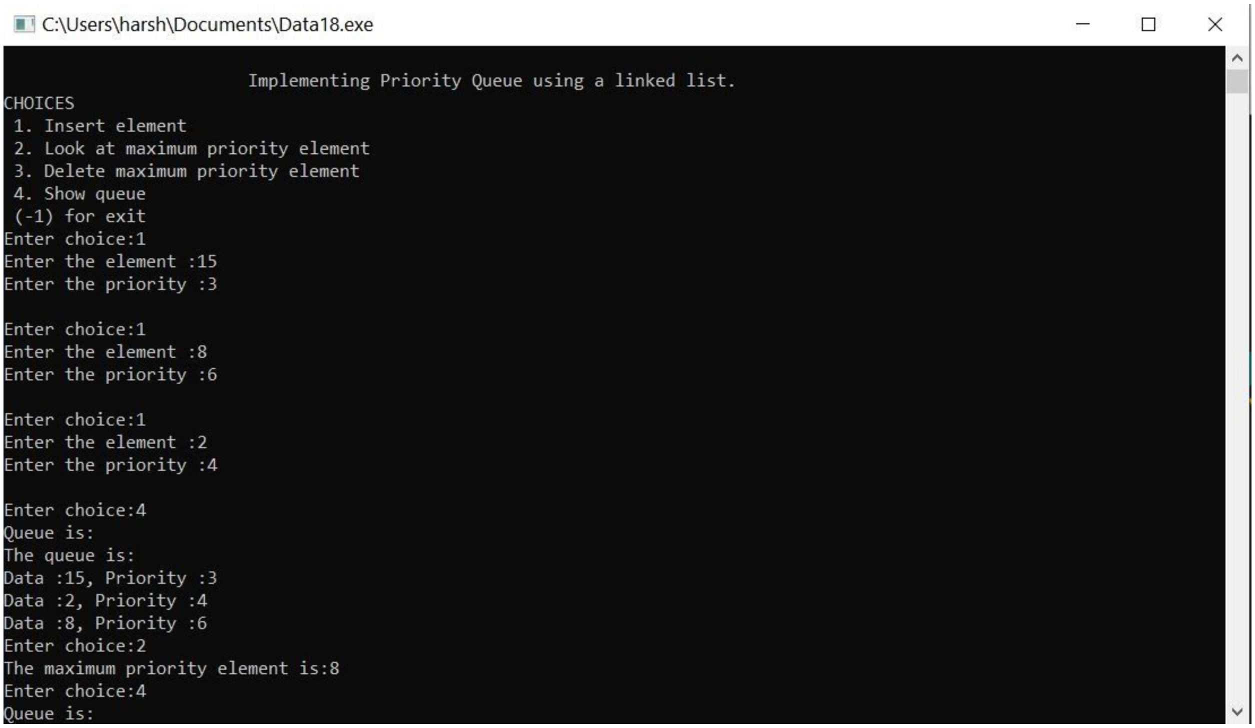
```
 #include<bits/stdc++.h>
using namespace std;
struct node
{
	int info;
	int priority;
	node *next;
	node *prev;
};

void insert(node **front,node **rear,int data, int p)
{
	node *temp=new node;
	temp->info=data;
	temp->priority=p;
	temp->next=temp->prev=NULL;
	if(*front==NULL)
		{ *front=*rear=temp;
		}
	else
	{
		if(p<= (*front)->priority)
		{
			temp->next=(*front);
			(*front)->prev=temp;
```

```
                (*front)=temp;
            }
            else if(p>= (*rear)->priority)
            {
                (*rear)->next=temp;
                temp->prev=(*rear);
                (*rear)=temp;
            }
            else
            {
                node *t=(*front);
                while(t->priority <= p)
                    t=t->next;
                temp->prev=t->prev;
                temp->next=t;
                t->prev->next=temp;
                t->prev=temp;
            }
    }
}
int peek(node **front, node **rear)
{
    if(*rear==NULL)
        return -1;
    else
        return ((*rear)->info);
}
int retMaxpriority(node **front,node **rear)
```

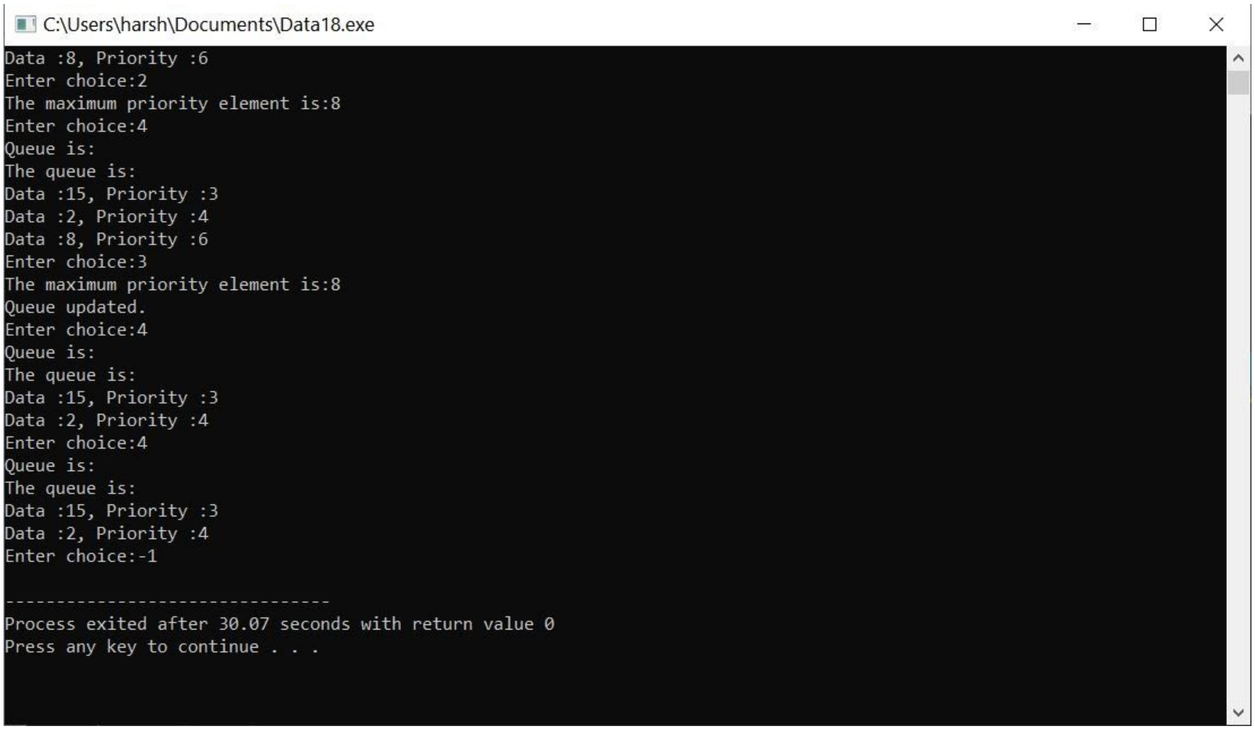
```
{
    if(*rear==NULL)   return -1;
    else
    {
        int dat= (*rear)->info;
        node *temp=*rear;
        (*rear)=(*rear)->prev;
        (*rear)->next=NULL;
        delete(temp);
        return(dat);
    }
}
void show(node **front, node **rear)
{
    if((*rear)==NULL)
        cout<<"\nEmpty queue.";
    else
    {   cout<<"\nThe queue is:";
        node *temp= (*front);
        while(temp!=NULL)
        {
            cout<<"\nData :"<<temp->info<<", Priority
:"<<temp->priority;
            temp=temp->next;
        }
    }
}
int main()
```

```
{
    node *front=NULL;
    node *rear=NULL;
    cout<<"\n\t\t\tImplementing Priority Queue using a
linked list.";
    cout<<"\nCHOICES\n 1. Insert element\n 2. Look at
maximum priority element\n 3. Delete maximum priority
element\n 4. Show queue\n (-1) for exit";
    int choice,elem,p;
    while(1)
    {
        cout<<"\nEnter choice:";
        cin>>choice;
        if(choice==-1)
            break;
        switch (choice)
        {
            case 1:{ cout<<"Enter the element :";
                     cin>>elem;
                     cout<<"Enter the priority :";
                     cin>>p;
                     insert(&front,&rear,elem,p);
                break;
            }
            case 2:{ cout<<"The maximum priority
element is:";
                    cout<<peek(&front,&rear);
                break;
                }
                case 3:{
                        cout<<"The maximum priority element
is:";
                        cout<<retMaxpriority(&front,&rear);
                        cout<<"\nQueue updated.";
                        break;
                }
                case 4:{
                        cout<<"Queue is:";
                        show(&front,&rear);
                        break;}
        }
    }
    return 0;}
```

- **OUTPUT**



```
                        Implementing Priority Queue using a linked list.
CHOICES
 1. Insert element
 2. Look at maximum priority element
 3. Delete maximum priority element
 4. Show queue
 (-1) for exit
Enter choice:1
Enter the element :15
Enter the priority :3

Enter choice:1
Enter the element :8
Enter the priority :6

Enter choice:1
Enter the element :2
Enter the priority :4

Enter choice:4
Queue is:
The queue is:
Data :15, Priority :3
Data :2, Priority :4
Data :8, Priority :6
Enter choice:2
The maximum priority element is:8
Enter choice:4
Queue is:
```



```
Data :8, Priority :6
Enter choice:2
The maximum priority element is:8
Enter choice:4
Queue is:
The queue is:
Data :15, Priority :3
Data :2, Priority :4
Data :8, Priority :6
Enter choice:3
The maximum priority element is:8
Queue updated.
Enter choice:4
Queue is:
The queue is:
Data :15, Priority :3
Data :2, Priority :4
Enter choice:4
Queue is:
The queue is:
Data :15, Priority :3
Data :2, Priority :4
Enter choice:-1

--------------------------------
Process exited after 30.07 seconds with return value 0
Press any key to continue . . .
```

19. WAP to implement a doubly ended queue using a linked list

- **C++ CODE**

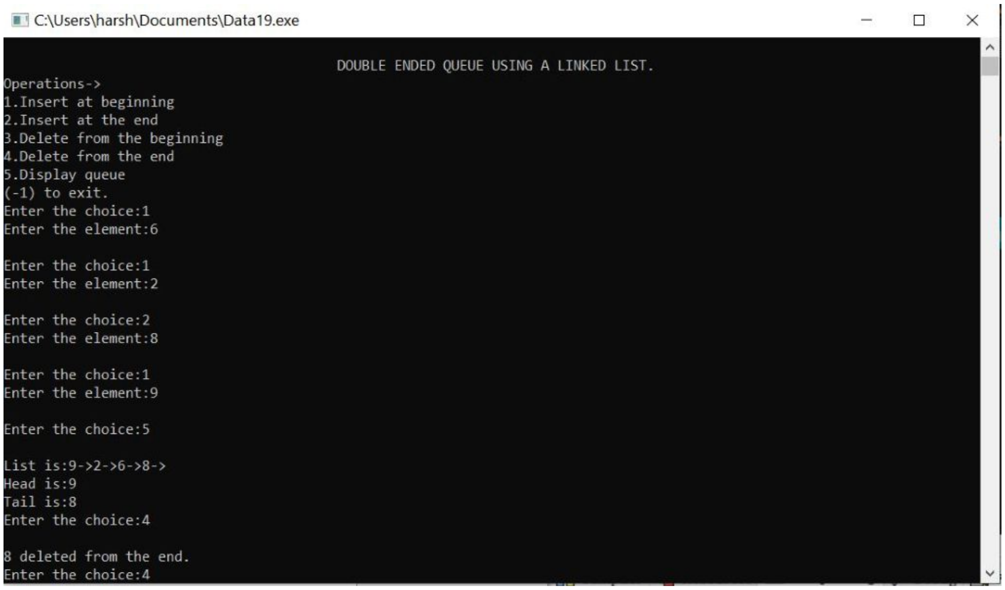
```
#include<iostream>
using namespace std;
struct node
{
	int data;
	struct node * next;
};
struct node* head;
struct node* tail;
class deq
{
	public: void insertB(int data)
		{
			node* temp;
			temp=new node;
			temp->data= data;
			if(head==NULL)
			{
			temp->next=NULL;
			head=tail=temp;}
			else
			{
			temp->next=head;
			head=temp;
			}
```

```
}

void insertE(int data)
{
node* temp;
temp=new node;
temp->data=data;
temp->next=NULL;
if(head==NULL)
    head=tail=temp;
else
{
    tail->next=temp;
    tail=temp;
}
}
void disp()
{
if(head==NULL)
   cout<<"No data to display...";
else
{
    node* temp;
    temp=head;
   cout<<"\nList is:";
    while (temp!=NULL)
    {cout<<temp->data<<"->";
    temp=temp->next;}
   cout<<"\nHead is:"<<head->data;
   cout<<"\nTail is:"<<tail->data;
}
}
int delE()
{
if(head==NULL)
    return -1;
else
 {  int d=tail->data;
    node* temp;
    temp=head;
    while(temp->next!=tail)
    temp=temp->next;
    node* tempb=tail;
    //important |v|
    temp->next=NULL;
    tail=temp;
    delete(tempb);
    return d;

}
}
int delB()
{
if(head==NULL)
    return -1;
    else
```

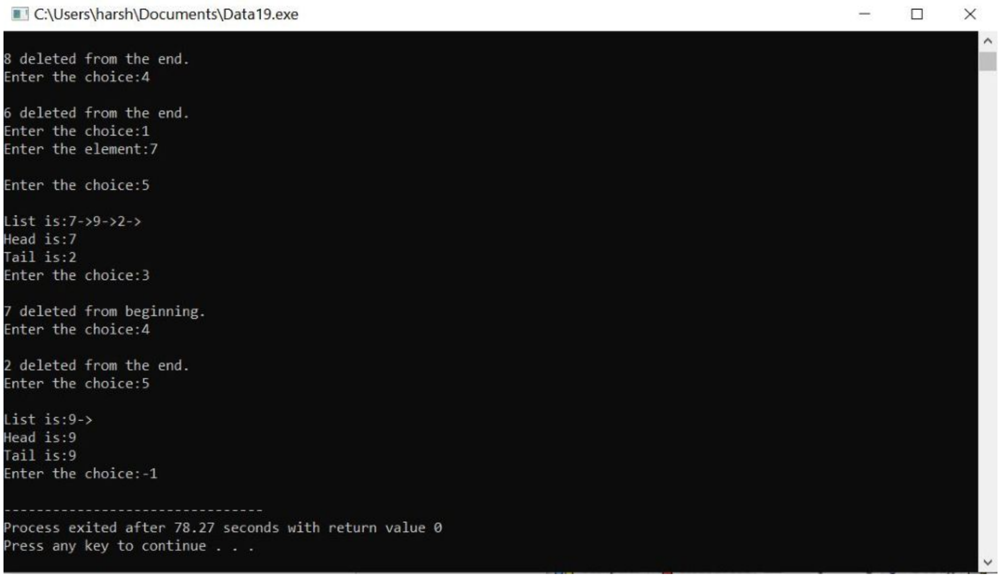
```
               {
               int dat= head->data;
               node* temp;
               temp=head;
               head=head->next;
               delete(temp);
               return dat;
               }
          }
};
int main()
{
   deq ob;
   cout<<"\n\t\t\t\t\t DOUBLE ENDED QUEUE USING A
LINKED LIST.";
   cout<<"\nOperations-> \n1.Insert at beginning\n2.Insert
at the end\n3.Delete from the beginning\n4.Delete from
the end\n5.Display queue\n(-1) to exit.";
     int choice;
     while(true)
     {
      cout<<"\nEnter the choice:";
      cin>>choice;
     if(choice==-1) break;
     else
     {
          switch (choice)
          {
           case(1):{ int ele;
                cout<<"Enter the element:";
                cin>>ele;
                ob.insertB(ele);
               break;
          }
           case(2):{ int elem;
                cout<<"Enter the element:";
                cin>>elem;
                ob.insertE(elem);
               break;
          }
           case(3):{ int ele=ob.delB();
                cout<<endl<<ele<<" deleted from
beginning.";
               break;
          }
           case(4):{ int ele=ob.delE();
                cout<<endl<<ele<<" deleted from the
end.";
               break;
          }
           case(5):{    ob.disp();
               break;
          }
           default:{
               break;
          }
```

```
            }
        }
        }

        return 0;
}
```

- **OUTPUT**



```
                                        DOUBLE ENDED QUEUE USING A LINKED LIST.
Operations->
1.Insert at beginning
2.Insert at the end
3.Delete from the beginning
4.Delete from the end
5.Display queue
(-1) to exit.
Enter the choice:1
Enter the element:6

Enter the choice:1
Enter the element:2

Enter the choice:2
Enter the element:8

Enter the choice:1
Enter the element:9

Enter the choice:5

List is:9->2->6->8->
Head is:9
Tail is:8
Enter the choice:4

8 deleted from the end.
Enter the choice:4
```



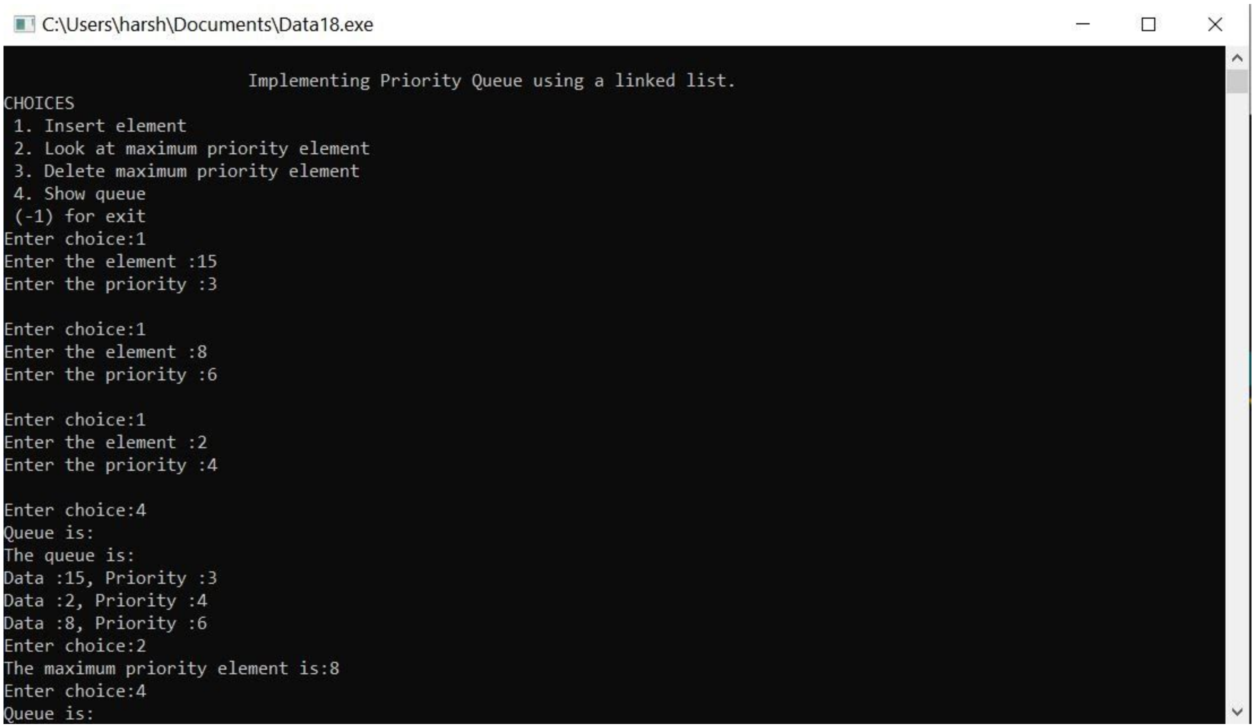
```
8 deleted from the end.
Enter the choice:4

6 deleted from the end.
Enter the choice:1
Enter the element:7

Enter the choice:5

List is:7->9->2->
Head is:7
Tail is:2
Enter the choice:3

7 deleted from beginning.
Enter the choice:4

2 deleted from the end.
Enter the choice:5

List is:9->
Head is:9
Tail is:9
Enter the choice:-1

--------------------------------
Process exited after 78.27 seconds with return value 0
Press any key to continue . . .
```

20. WAP to construct a Binary Tree and display it’s preorder, inorder, postorder traversals.

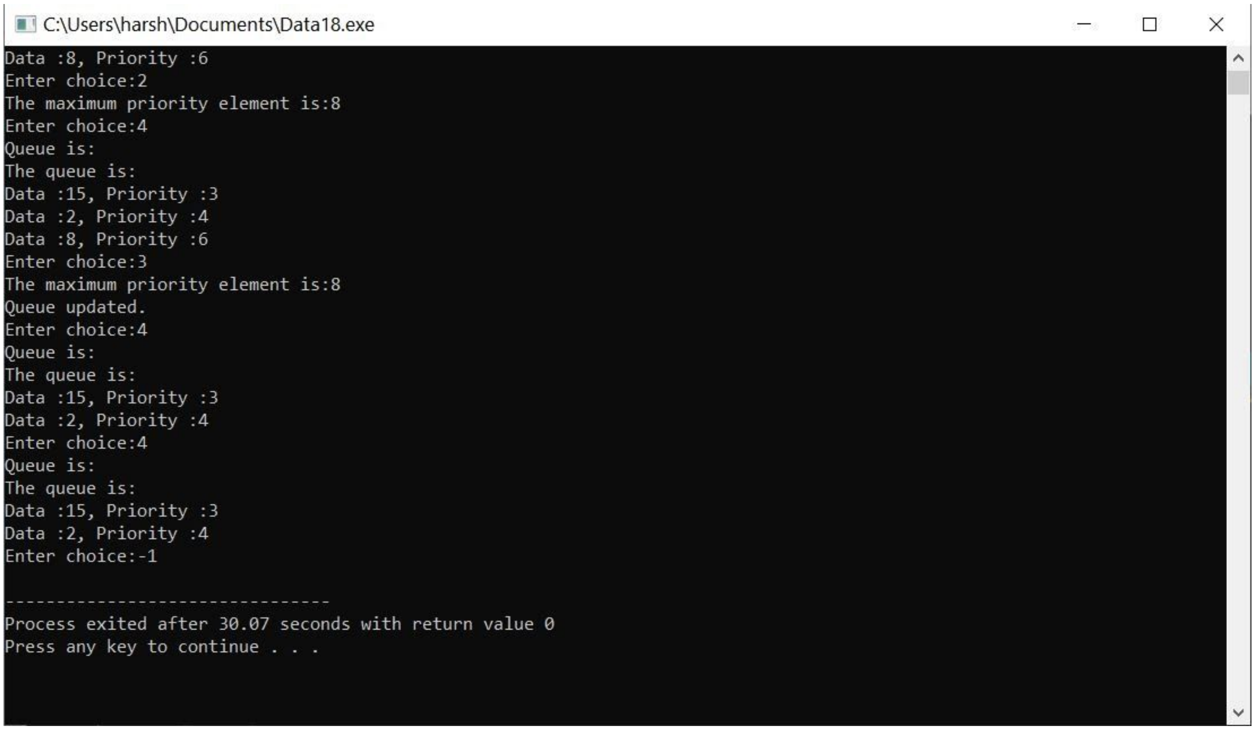
```
#include<bits/stdc++.h>
#include<iostream>
using namespace std;
struct bnode
{
	int data;
	bnode *left,*right;
};

bnode * insert(bnode *root, int dat)
{
	if(root==NULL)
		{
			bnode *temp= new bnode;
			temp->data=dat;
			temp->left=temp->right=NULL;
			root=temp;
		}
	else
	{
		if(dat<=root->data)
			root->left=insert(root->left,dat);

		else
			root->right=insert(root->right,dat);
		}
		return root;
}
void preorder(bnode *root)
{
		if(root==NULL)	return;
		else
		{
			cout<<root->data<<" ";
			preorder(root->left);
			preorder(root->right);
		}
}
void inorder(bnode *root)
{
		if(root==NULL)	return;
		else
		{
			inorder(root->left);
			cout<<root->data<<" ";
			inorder(root->right);
		}
}
void postorder(bnode *root)
{
		if(root==NULL)	return;
		else
```

```
    {
        postorder(root->left);
        postorder(root->right);
        cout<<root->data<<" ";
    }
}

int main()
{   bnode *root;
    root=NULL;
  int ch,a,b;
    cout<<"\n\t\t\t\tProgram to implement Binary
Tree...\n\nPress\n1 For insertion\n";
    cout<<"2. Preorder Traversal\n3. Inorder traversal";
    cout<<" \n4. Postorder traversal\n(-1) for exit.";
    while(1)
    {
    k:cout<<"\nEnter the choice:";
    cin>>ch;
    if(ch==-1)
    break;
    else
    {
    switch(ch)
    {
        case 1: cout<<"Enter the data to be inserted:";
                cin>>a;
                root=insert(root,a);
                cout<<endl<<a<<" inserted.";
                break;
        case 2:cout<<"\nThe preorder traversal of tree
is:";
                preorder(root);
                break;
        case 3: {cout<<"\nThe inorder traversal of tree
is:";
                inorder(root);
                break;}
        case 4:{cout<<"\nThe postorder traversal of tree
is:";
                postorder(root);
                break;}

        default:cout<<"\nWrong choice.";
                goto k;

    }

}
}
return 0;
}
```

- **OUTPUT**



```
                        Implementing Priority Queue using a linked list.
CHOICES
 1. Insert element
 2. Look at maximum priority element
 3. Delete maximum priority element
 4. Show queue
 (-1) for exit
Enter choice:1
Enter the element :15
Enter the priority :3

Enter choice:1
Enter the element :8
Enter the priority :6

Enter choice:1
Enter the element :2
Enter the priority :4

Enter choice:4
Queue is:
The queue is:
Data :15, Priority :3
Data :2, Priority :4
Data :8, Priority :6
Enter choice:2
The maximum priority element is:8
Enter choice:4
Queue is:
```



```
Data :8, Priority :6
Enter choice:2
The maximum priority element is:8
Enter choice:4
Queue is:
The queue is:
Data :15, Priority :3
Data :2, Priority :4
Data :8, Priority :6
Enter choice:3
The maximum priority element is:8
Queue updated.
Enter choice:4
Queue is:
The queue is:
Data :15, Priority :3
Data :2, Priority :4
Enter choice:4
Queue is:
The queue is:
Data :15, Priority :3
Data :2, Priority :4
Enter choice:-1

--------------------------------
Process exited after 30.07 seconds with return value 0
Press any key to continue . . .
```

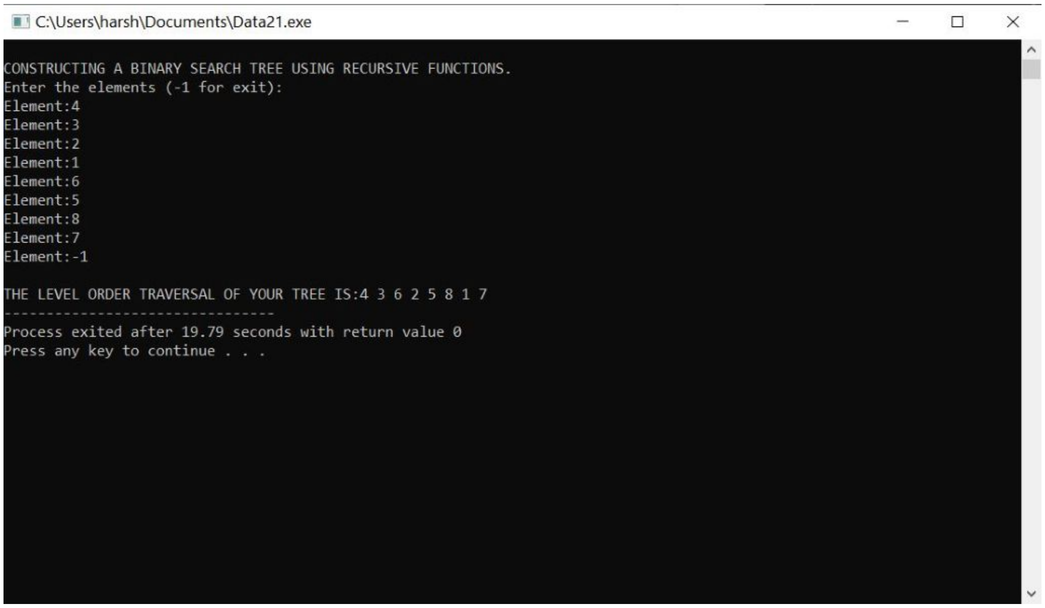
21. WAP to construct a Binary Search Tree

```
#include<iostream>
#include<queue>
using namespace std;
struct bnode
{
      int data;
   bnode *left,*right;
};

bnode * insert(bnode *root, int dat)
{
      if(root==NULL)
      {
          bnode *temp= new bnode;
            temp->data=dat;
            temp->left=temp->right=NULL;
            root=temp;
      }
      else
      {
      if(dat<=root->data)
            root->left=insert(root->left,dat);

      else
            root->right=insert(root->right,dat);

      }
```

```
    return root;
}
void BFStree(bnode *root)
{
    if(root==NULL)
     cout<<"\nTree is empty";
    else
  {   queue<bnode*> Q; //stl queue inbuilt function.
Making a queue of addresses.
     Q.push(root);
    while(!Q.empty())
    {
       bnode *current=Q.front(); // this just returns the
address of the front element of the queue but
                    //  doesn't pop it from the queue.
        if(current->left!=NULL)
         Q.push(current->left);
        if(current->right!=NULL)
         Q.push(current->right);
       cout<<Q.front()->data<<" ";
       Q.pop(); // to pop the front element from the queue.
    }
    }
}
int main()
{
  bnode *root;
    root=NULL;
```

```
      int ele;
    cout<<"\nCONSTRUCTING A BINARY SEARCH TREE
USING RECURSIVE FUNCTIONS.";
    cout<<"\nEnter the elements (-1 for exit):\n";
    while(1)
      {
       cout<<"Element:";
       cin>>ele;
      if(ele==-1) break;
      else
            root=insert(root,ele);
      }
    cout<<"\nTHE LEVEL ORDER TRAVERSAL OF YOUR
TREE IS:";
    BFStree(root);
return 0;
}
```

- **OUTPUT**



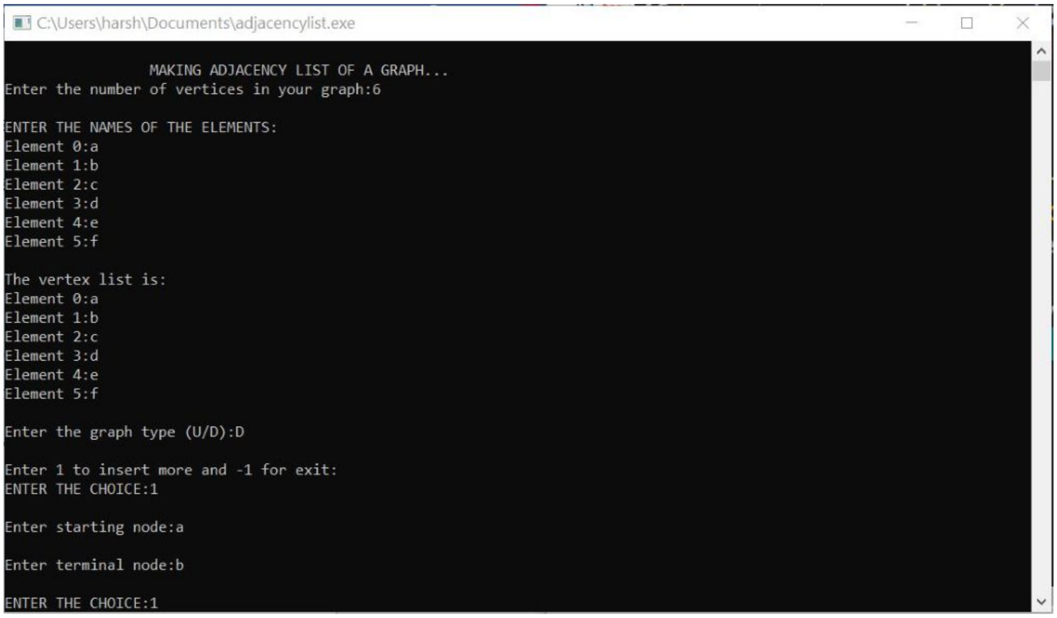
```

CONSTRUCTING A BINARY SEARCH TREE USING RECURSIVE FUNCTIONS.
Enter the elements (-1 for exit):
Element:4
Element:3
Element:2
Element:1
Element:6
Element:5
Element:8
Element:7
Element:-1

THE LEVEL ORDER TRAVERSAL OF YOUR TREE IS:4 3 6 2 5 8 1 7
--------------------------------
Process exited after 19.79 seconds with return value 0
Press any key to continue . . .
```

22. WAP to construct a graph (adjacency lists used)

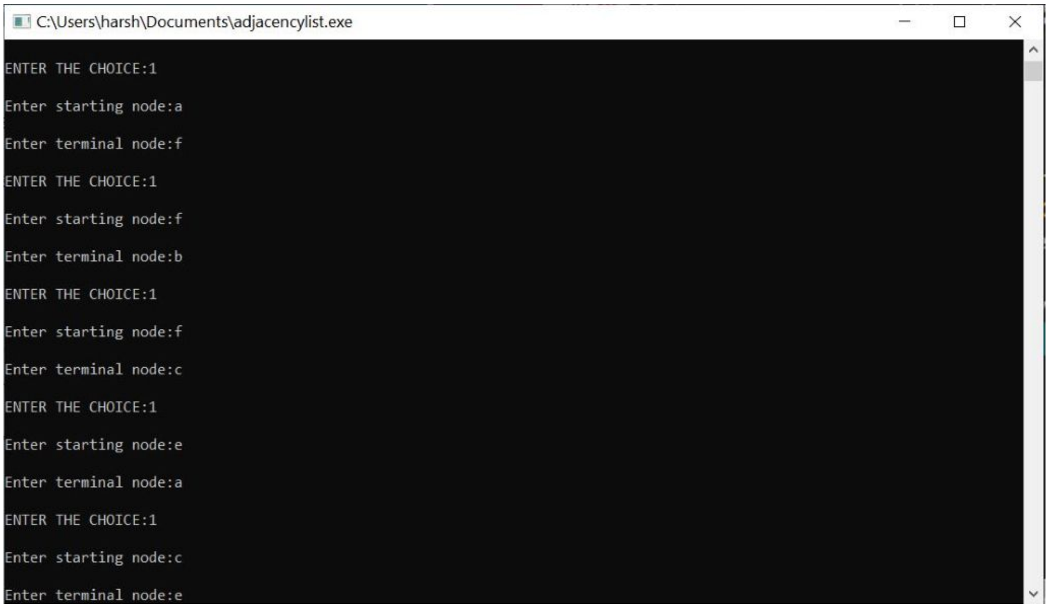
```
#include<bits/stdc++.h>
#define For(i,n) for(int i=0;i<n;i++)
#define tr(v,it) for(typeof(v.begin())
it=v.begin();it!=v.end();it++)
using namespace std;
int main()
{    string ele;

    cout<<"\n\t\t MAKING ADJACENCY LIST OF A
GRAPH...";
    cout<<"\nEnter the number of vertices in your graph:";
       int n;
    cin>>n;
    cout<<"\nENTER THE NAMES OF THE
ELEMENTS:\n";
       vector< string > v;
       For(i,n)
       { cout<<"Element "<<i<<":";
     cin>>ele;
      v.push_back(ele);
       }
    cout<<"\nThe vertex list is:\n";
       For(i,n)
       { cout<<"Element "<<i<<":"<<v[i]<<endl;
       }
       char type;
    cout<<"\nEnter the graph type (U/D):";
    cin>>type;
       vector <int> edge[n];
       string v1,v2;
    cout<<"\nEnter 1 to insert more and -1 for exit:";
       int elem,i,j;
       while(true)
    {    cout<<"\nENTER THE CHOICE:";
       cin>>elem;
       if(elem==-1) break;
       cout<<"\nEnter starting node:";
       cin>>v1;
       cout<<"\nEnter terminal node:";
       cin>>v2;
        i= find(v.begin(),v.end(),v1)-v.begin();
       j= find(v.begin(),v.end(),v2)-v.begin();
       if(type=='D')
              edge[i].push_back(j);
       else
       {

if(find(edge[i].begin(),edge[i].end(),j)==edge[i].end())
              edge[i].push_back(j);

if(find(edge[j].begin(),edge[j].end(),i)==edge[j].end())
              edge[j].push_back(i);
       }
       }
    for(int i=0;i<n;i++)
```

```
{    cout<<"\nVertex "<<i<<", "<<v[i]<<" connnections:";
    for(int j=0;j<edge[i].size();j++)
          {
            cout<<v[edge[i][j]]<<" ";
          }
    cout<<endl;
}      return 0;
   }
```

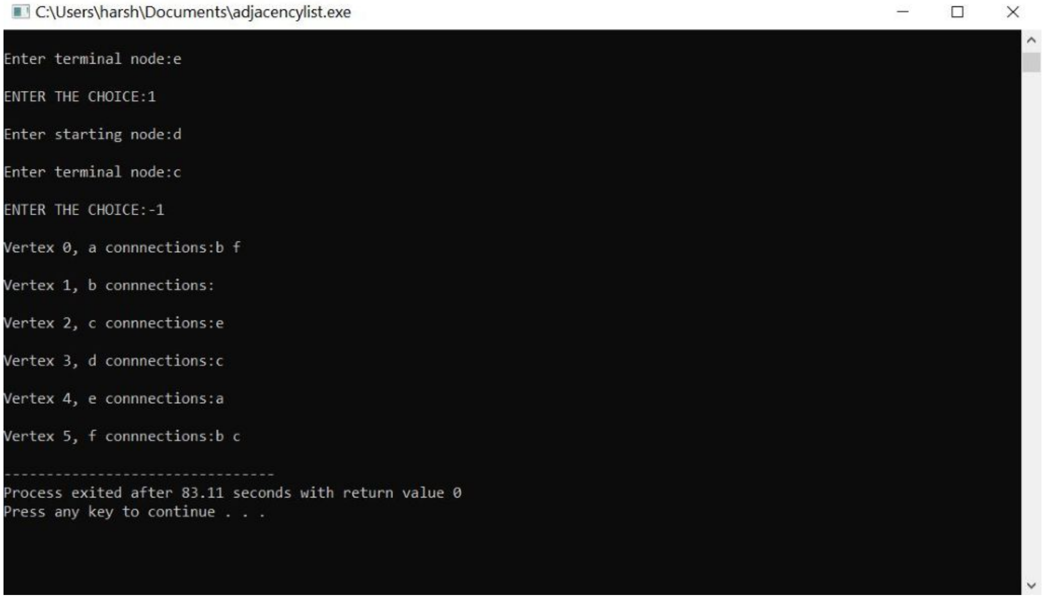
- **OUTPUT**



```
                 MAKING ADJACENCY LIST OF A GRAPH...
Enter the number of vertices in your graph:6

ENTER THE NAMES OF THE ELEMENTS:
Element 0:a
Element 1:b
Element 2:c
Element 3:d
Element 4:e
Element 5:f

The vertex list is:
Element 0:a
Element 1:b
Element 2:c
Element 3:d
Element 4:e
Element 5:f

Enter the graph type (U/D):D

Enter 1 to insert more and -1 for exit:
ENTER THE CHOICE:1

Enter starting node:a

Enter terminal node:b

ENTER THE CHOICE:1
```



```
ENTER THE CHOICE:1

Enter starting node:a

Enter terminal node:f

ENTER THE CHOICE:1

Enter starting node:f

Enter terminal node:b

ENTER THE CHOICE:1

Enter starting node:f

Enter terminal node:c

ENTER THE CHOICE:1

Enter starting node:e

Enter terminal node:a

ENTER THE CHOICE:1

Enter starting node:c

Enter terminal node:e
```



```
Enter terminal node:e

ENTER THE CHOICE:1

Enter starting node:d

Enter terminal node:c

ENTER THE CHOICE:-1

Vertex 0, a connnections:b f

Vertex 1, b connnections:

Vertex 2, c connnections:e

Vertex 3, d connnections:c

Vertex 4, e connnections:a

Vertex 5, f connnections:b c

--------------------------------
Process exited after 83.11 seconds with return value 0
Press any key to continue . . .
```

23. WAP to calculate the distance between two vertices in a graph

- **C++ CODE**

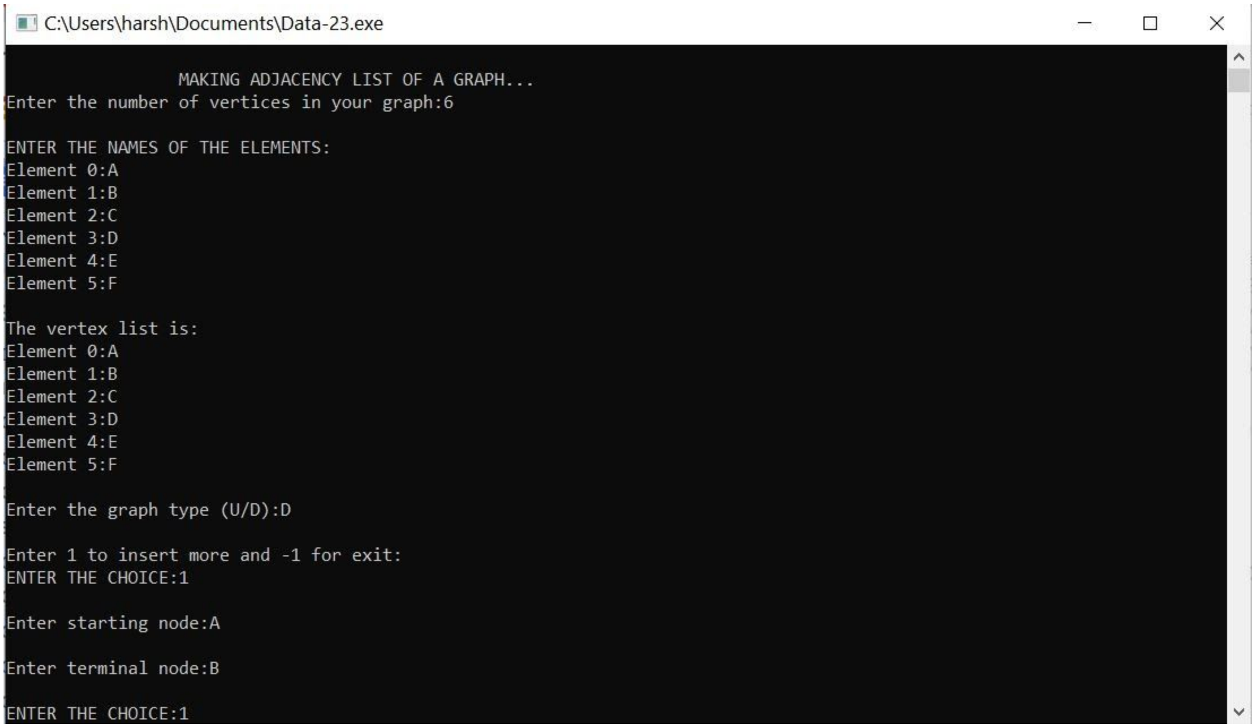
```
#include<bits/stdc++.h>
#define For(i,n) for(int i=0;i<n;i++)
#define tr(v,it) for(typeof(v.begin()) it=v.begin();it!=v.end();it++)
using namespace std;
int path(int i,int j,vector <int> edges[],int n)
{   queue <int> Q;
    Q.push(i);
    bool visited[n]={false};
    visited[i]=true;
    map <int,int> dist;
    dist[i]=0;
    while(visited[j]!=true && !Q.empty())
    {
        int current= Q.front();
        int pdist=dist[current];// the distance of the current
node from the source node
        for(int m=0;m<edges[current].size();m+=1)
        {   if(!visited[edges[current][m]])
             {
              Q.push(edges[current][m]);
             visited[edges[current][m]]=true;
             dist[edges[current][m]]=pdist+1;//distance of
current node+1 (as it is child)
             }
        }
```

```
              Q.pop();
              }
       if(visited[j]==false)
              return -1;
       else
              return dist[j];
}
int main()
{      string ele;

       cout<<"\n\t\t MAKING ADJACENCY LIST OF A GRAPH...";
       cout<<"\nEnter the number of vertices in your graph:";
       int n;
       cin>>n;
       cout<<"\nENTER THE NAMES OF THE ELEMENTS:\n";
       vector< string > v;
       For(i,n)
       { cout<<"Element "<<i<<":";
        cin>>ele;
         v.push_back(ele);
       }
       cout<<"\nThe vertex list is:\n";
       For(i,n)
       { cout<<"Element "<<i<<":"<<v[i]<<endl;
       }
       char type;
       cout<<"\nEnter the graph type (U/D):";
```

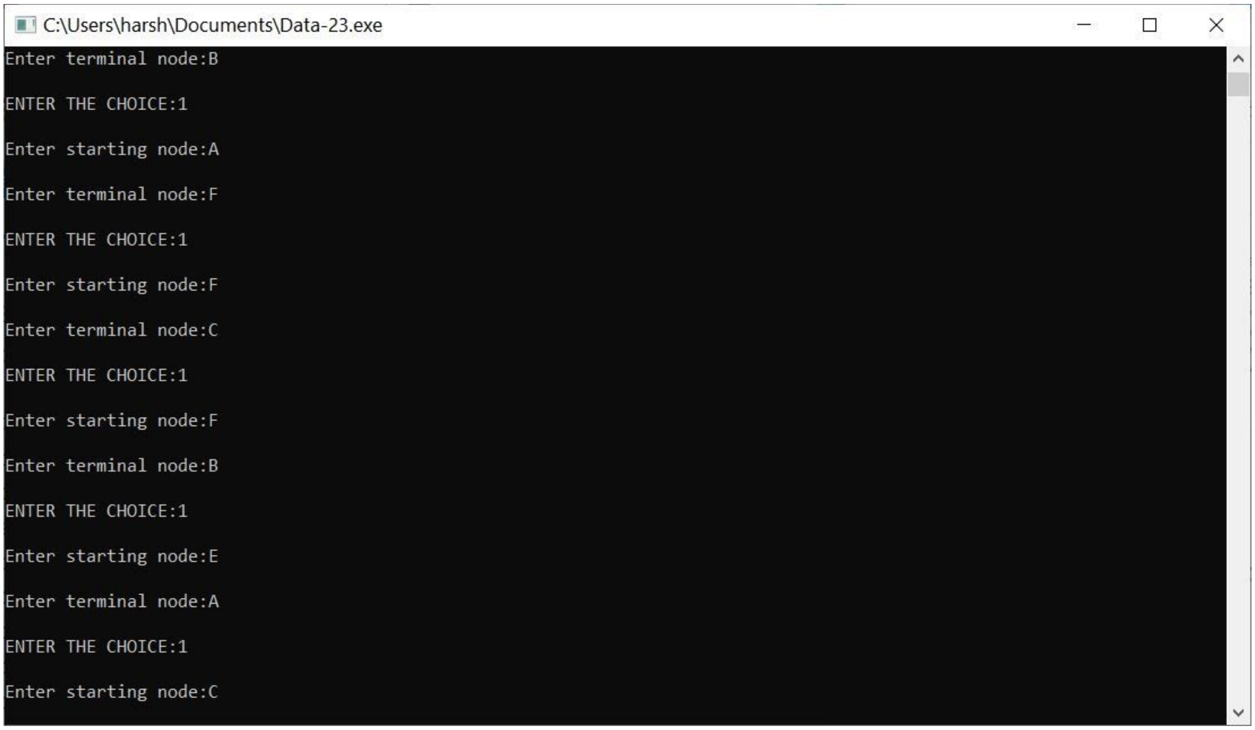
```
    cin>>type;
    vector <int> edge[n];
    string v1,v2;
    cout<<"\nEnter 1 to insert more and -1 for exit:";
    int elem,i,j;
    while(true)
    {   cout<<"\nENTER THE CHOICE:";
        cin>>elem;
        if(elem==-1) break;
        cout<<"\nEnter starting node:";
        cin>>v1;
        cout<<"\nEnter terminal node:";
        cin>>v2;
         i= find(v.begin(),v.end(),v1)-v.begin();
         j= find(v.begin(),v.end(),v2)-v.begin();
        if(type=='D')
            edge[i].push_back(j);
        else
        {

if(find(edge[i].begin(),edge[i].end(),j)==edge[i].end())
                edge[i].push_back(j);

if(find(edge[j].begin(),edge[j].end(),i)==edge[j].end())
                edge[j].push_back(i);
        }
    }
    for(int i=0;i<n;i++)
```

```
	{	cout<<"\nVertex "<<i<<", "<<v[i]<<" connections:";
		for(int j=0;j<edge[i].size();j++)
			{
				cout<<v[edge[i][j]]<<" ";
			}
		cout<<endl;
	}
	cout<<"\nFINDING THE PATH LENGTH BETWEEN TWO
VERTICES\n";
	cout<<"\nEnter starting node:";
	cin>>v1;
	cout<<"\nEnter terminal node:";	cin>>v2;
	i= find(v.begin(),v.end(),v1)-v.begin();
	j= find(v.begin(),v.end(),v2)-v.begin();
	int length;
	length=path(i,j,edge,n);
	if(length==-1)
		cout<<"\nNo path exists.";
	else
		cout<<"\nLength of path between "<<v1<<" and
"<<v2<<" is: "<<length;
		return 0;
	}
```

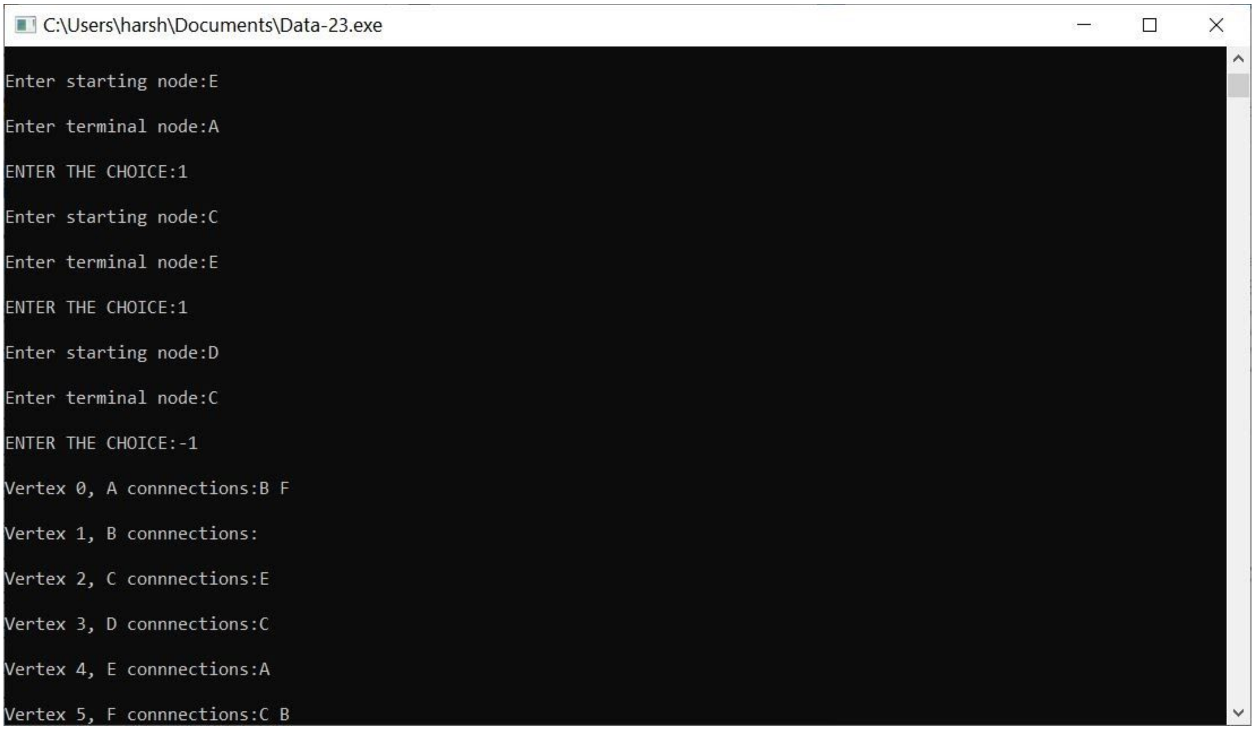
- **OUTPUT**



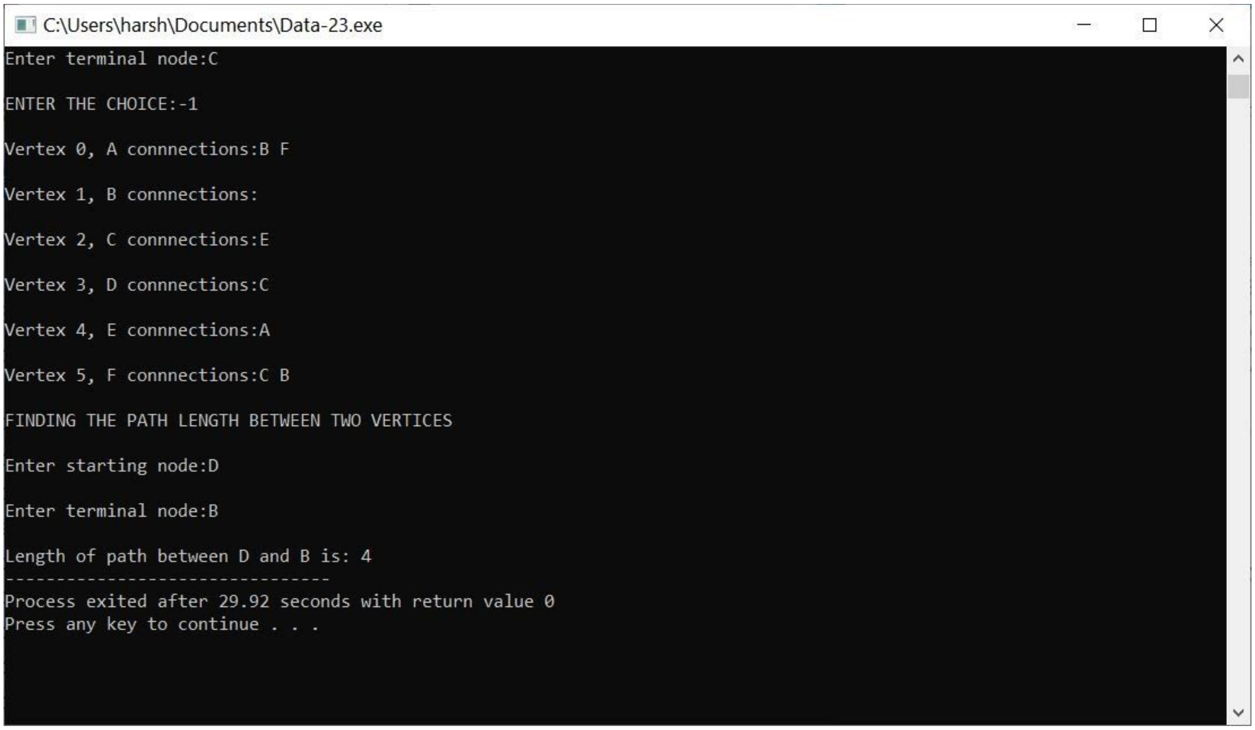
```
                 MAKING ADJACENCY LIST OF A GRAPH...
Enter the number of vertices in your graph:6

ENTER THE NAMES OF THE ELEMENTS:
Element 0:A
Element 1:B
Element 2:C
Element 3:D
Element 4:E
Element 5:F

The vertex list is:
Element 0:A
Element 1:B
Element 2:C
Element 3:D
Element 4:E
Element 5:F

Enter the graph type (U/D):D

Enter 1 to insert more and -1 for exit:
ENTER THE CHOICE:1

Enter starting node:A

Enter terminal node:B

ENTER THE CHOICE:1
```



```
Enter terminal node:B

ENTER THE CHOICE:1

Enter starting node:A

Enter terminal node:F

ENTER THE CHOICE:1

Enter starting node:F

Enter terminal node:C

ENTER THE CHOICE:1

Enter starting node:F

Enter terminal node:B

ENTER THE CHOICE:1

Enter starting node:E

Enter terminal node:A

ENTER THE CHOICE:1

Enter starting node:C
```



```
Enter starting node:E

Enter terminal node:A

ENTER THE CHOICE:1

Enter starting node:C

Enter terminal node:E

ENTER THE CHOICE:1

Enter starting node:D

Enter terminal node:C

ENTER THE CHOICE:-1

Vertex 0, A connnections:B F

Vertex 1, B connnections:

Vertex 2, C connnections:E

Vertex 3, D connnections:C

Vertex 4, E connnections:A

Vertex 5, F connnections:C B
```



```
Enter terminal node:C

ENTER THE CHOICE:-1

Vertex 0, A connnections:B F

Vertex 1, B connnections:

Vertex 2, C connnections:E

Vertex 3, D connnections:C

Vertex 4, E connnections:A

Vertex 5, F connnections:C B

FINDING THE PATH LENGTH BETWEEN TWO VERTICES

Enter starting node:D

Enter terminal node:B

Length of path between D and B is: 4
--------------------------------
Process exited after 29.92 seconds with return value 0
Press any key to continue . . .
```

24. WAP to calculate the distances between every pairs of vertices in a graph

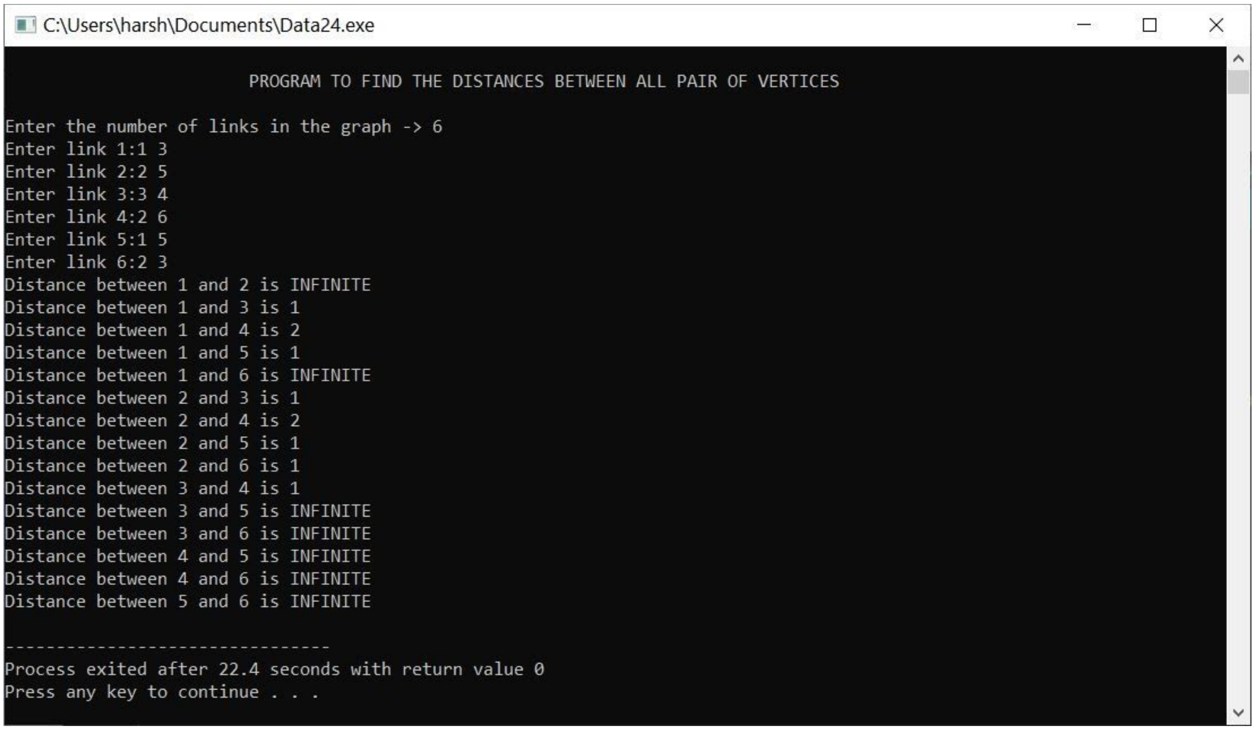
- **C++ CODE**

```
#include<iostream>
#include<map>
#include<list>
#include<queue>
#include<set>
#include<climits>
using namespace std;
class graph{
public:
    map<int,list<int > > adjlist;
    set<int> s;
    void addedge(int u, int v)
    {
        s.insert(u);
        s.insert(v);
        adjlist[u].push_back(v);
    }
    void print()
    {
        for(map<int,list<int> >:: iterator
it=adjlist.begin();it!=adjlist.end();it++)
    {
        cout<<it->first<<" -> ";
        for(list<int>::iterator
it1=adjlist[it->first].begin();it1!=adjlist[it->first].end();it1++)
```

```
        {
            cout<<*it1<<" ";
        }
        cout<<endl;
    }
    }
    void distances(int src)
    {
        map<int,int> dist;
        for(set<int>::iterator it=s.begin();it!=s.end();it++)
            dist[*it]=INT_MAX;
        map<int,bool> visited;
        queue<int> q;
        q.push(src);
        dist[src]=0;
        visited[src]=0;
        while(!q.empty())
        {
            int node=q.front();
            q.pop();
            int pdist=dist[node];
            for(list<int>::iterator
it=adjlist[node].begin();it!=adjlist[node].end();it++)
            {
                if(!visited[*it])
                {
                    dist[*it]=pdist+1;
                    visited[*it]=true;
                    q.push(*it);
                }
            }
        }
        cout<<"The distances of all nodes from "<<src<<" are->
"<<endl;
        for(map<int,int>::iterator it=dist.begin();it!=dist.end();it++){

            cout<<it->first<<" -> ";
            if(it->second != INT_MAX)
                cout<<it->second<<endl;
            else
                cout<<"NOT REACHABLE"<<endl;
        }
    }
    int distance(int src, int dest)
    {
        int answer=INT_MAX;
        queue<int> q;
        q.push(src);
        map<int,int> dist;
        map<int,bool> visited;
        visited[src]=true;
        dist[src]=0;

        while(!q.empty())
        {
            int node=q.front();
```

```
          q.pop();
          int pdist=dist[node];
          for(list<int>::iterator
it=adjlist[node].begin();it!=adjlist[node].end();it++)
            {
               if(*it == dest)
                  return (pdist+1);
               if(!visited[*it])
               {
                  dist[*it]=pdist+1;
                  visited[*it]=true;
                  q.push(*it);
               }

            }

        }
        return answer;
    }
};
int main()
{     cout<<"\n\t\t\tPROGRAM TO FIND THE DISTANCES
BETWEEN ALL PAIR OF VERTICES\n";
    graph g;
    cout<<"\nEnter the number of links in the graph -> ";
    int n;
    cin>>n;
    int u,v,i;
    i=1;
    while(n--)
    {  cout<<"Enter link "<<i<<":";
        cin>>u>>v;
        g.addedge(u,v);
        i++;
    }
    //g.print();
    vector<int> elements;
    for(set<int> ::iterator it=g.s.begin();it!=g.s.end();it++)
        elements.push_back(*it);
    for(int i=0;i<elements.size();i++)
    {
        for(int j=i+1;j<elements.size();j++)
        {
            cout<<"Distance between "<<elements[i]<<" and 
"<<elements[j]<<" is ";
            int d=g.distance(elements[i],elements[j]);
            if(d==INT_MAX)
                cout<<"INFINITE"<<endl;
            else
                cout<<d<<endl;

        }
    }

    return 0;
}
```

- **OUTPUT**



```
                        PROGRAM TO FIND THE DISTANCES BETWEEN ALL PAIR OF VERTICES

Enter the number of links in the graph -> 6
Enter link 1:1 3
Enter link 2:2 5
Enter link 3:3 4
Enter link 4:2 6
Enter link 5:1 5
Enter link 6:2 3
Distance between 1 and 2 is INFINITE
Distance between 1 and 3 is 1
Distance between 1 and 4 is 2
Distance between 1 and 5 is 1
Distance between 1 and 6 is INFINITE
Distance between 2 and 3 is 1
Distance between 2 and 4 is 2
Distance between 2 and 5 is 1
Distance between 2 and 6 is 1
Distance between 3 and 4 is 1
Distance between 3 and 5 is INFINITE
Distance between 3 and 6 is INFINITE
Distance between 4 and 5 is INFINITE
Distance between 4 and 6 is INFINITE
Distance between 5 and 6 is INFINITE

--------------------------------
Process exited after 22.4 seconds with return value 0
Press any key to continue . . .
```

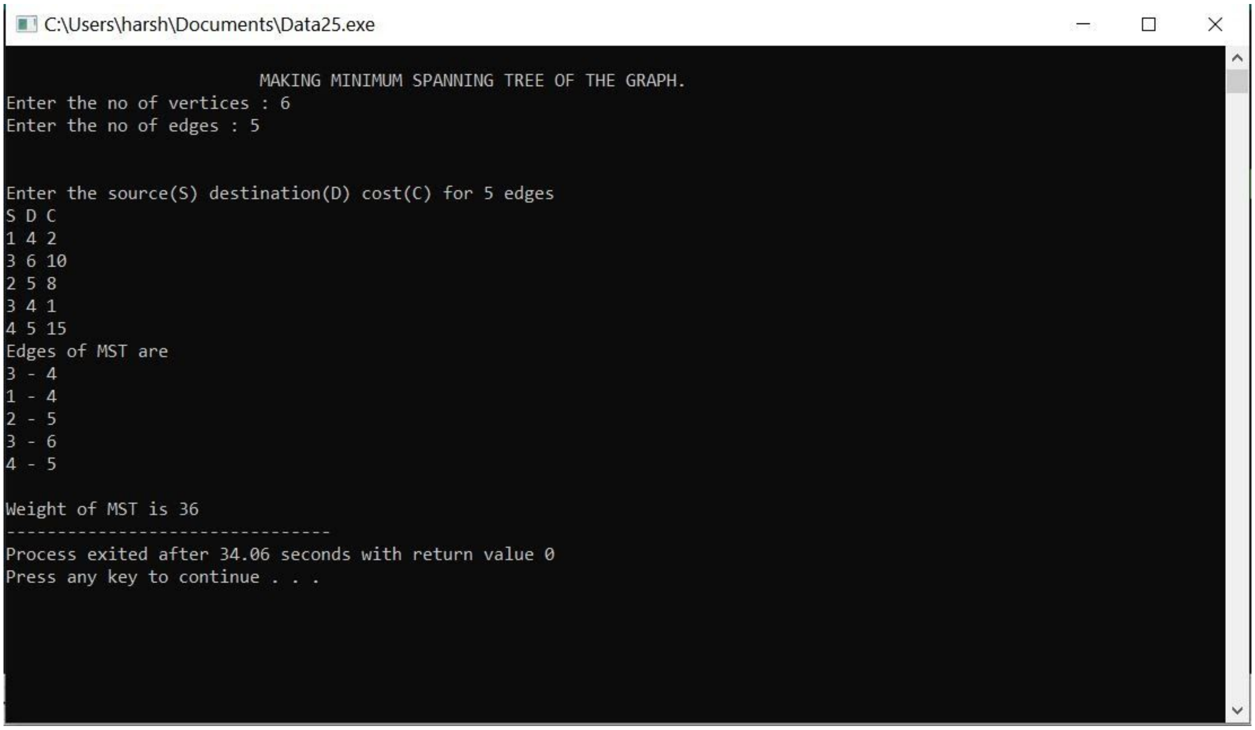
25. WAP to construct a minimal spanning tree of a graph

- **C++ CODE**

```
#include<bits/stdc++.h>
#include<iostream>
using namespace std;

typedef  pair<int, int> iPair;
struct Graph
{
   int V, E;
   vector< pair<int, iPair> > edges;
   Graph(int V, int E)
   {
      this->V = V;
      this->E = E;
   }
   void addEdge(int u, int v, int w)
   {
      edges.push_back({w, {u, v}});
   }



   int kruskalMST();
};

struct DisjointSets
{
   int *parent, *rnk;
int n;
DisjointSets(int n)
{
    this->n = n;
    parent = new int[n+1];
    rnk = new int[n+1];
    for (int i = 0; i <= n; i++)
    {
        rnk[i] = 0;
        parent[i] = i;
    }
}
int find(int u)
{
    if (u != parent[u])
        parent[u] = find(parent[u]);
    return parent[u];
}


void merge(int x, int y)
{
    x = find(x), y = find(y);
    if (rnk[x] > rnk[y])
        parent[y] = x;
    else
        parent[x] = y;
        if (rnk[x] == rnk[y])
            rnk[y]++;
    }
};

int Graph::kruskalMST()
{
    int mst_wt = 0;
    sort(edges.begin(), edges.end());
    DisjointSets ds(V);
    vector< pair<int, iPair> >::iterator it;
    for (it=edges.begin(); it!=edges.end(); it++)
    {
        int u = it->second.first;
        int v = it->second.second;
        int set_u = ds.find(u);
        int set_v = ds.find(v);
        if (set_u != set_v)
        {
            cout << u << " - " << v << endl;
            mst_wt += it->first;
            ds.merge(set_u, set_v);
        }
    }
return mst_wt;
}

int main()
```

```
{     cout<<"\n\t\t\t MAKING MINIMUM SPANNING TREE OF
THE GRAPH.";
    int V , E ;
    cout<< "\nEnter the no of vertices : " ;
    cin >> V ;
    cout<<"Enter the no of edges : ";
    cin>> E;
    Graph g(V, E);
    cout<<"\n\n";
   int x,y,z;
   cout<< "Enter the source(S) destination(D) cost(C) for
"<<E<<" edges"<<endl;
   cout<<"S D C"<<endl;
   for(int i=0 ;i<E ;i++)
   {
       cin>> x >> y >> z ;
       g.addEdge(x, y, z);
   }
    cout << "Edges of MST are \n";
    int mst_wt = g.kruskalMST();
    cout << "\nWeight of MST is " << mst_wt;

    return 0;
}
```

- **OUTPUT**



```
                         MAKING MINIMUM SPANNING TREE OF THE GRAPH.
Enter the no of vertices : 6
Enter the no of edges : 5


Enter the source(S) destination(D) cost(C) for 5 edges
S D C
1 4 2
3 6 10
2 5 8
3 4 1
4 5 15
Edges of MST are
3 - 4
1 - 4
2 - 5
3 - 6
4 - 5

Weight of MST is 36
--------------------------------
Process exited after 34.06 seconds with return value 0
Press any key to continue . . .
```