**Implement a program to check if the link list is a palindrome**

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node

{

int data;

struct node \*next;

}

\*list=NULL,\*p,\*s,\*q,\*r; //\*p is used for new node

class LinkPal

{

public:

int choice,value;

void get()

{

do

{

cout<<"0.Exit\n1.Add Elemnet \n2.Check if Pallindrome \n3.Display \n";

cout<<"Enter Your Choice : "<<" ";

cin>>choice;

switch(choice)

{

case 0:

break;

case 1:

insert();

break;

case 2:

check();

break;

case 3:

display();

break;

default:

cout<<"invalid input"<<endl<<endl;

}

}while(choice!=0);

}

void insert()

{

cout<<"Enter the value : ";

cin>>value;

p=(struct node\*)malloc(sizeof(node));

p->data=value;

if(list == NULL)

{

p->next=NULL;

list=p;

display();

}

else

{

q=list;

while(q->next != NULL)

{

q=q->next;

}

q->next=p;

p->next=NULL;

display();

}

}

int count\_ele()

{

int c=0;

p=list;

while(p != NULL)

{

p=p->next;

c++;

}

return c;

// cout<<"The Number of Elements is : "<<c<<endl<<endl;

}

void check()

{

if(list == NULL)

{

cout<<endl<<"List is Empty "<<endl<<endl;

}

else

{

int flag = count\_ele()/2;

p=list;

q=NULL;

while(flag > 0)

{

r=list;

while(r->next != q)

{

r=r->next;

}

if(p->data == r->data)

{

p = p->next;

q=r;

flag--;

}

else

flag = -1;

}

if(flag == -1)

cout<<"The List is a Pallindrome";

else

cout<<"The List is not a Pallindrome";

}

cout<<endl<<endl;

}

void display()

{

if(list==NULL)

{

cout<<endl<<"List is Empty "<<endl<<endl;

}

else

{

cout<<"The List is : ";

q=list;

while(q !=NULL)

{

cout<<q->data<<"|----->";

q=q->next;

}

cout<<endl<<endl;

}

}

};

int main()

{

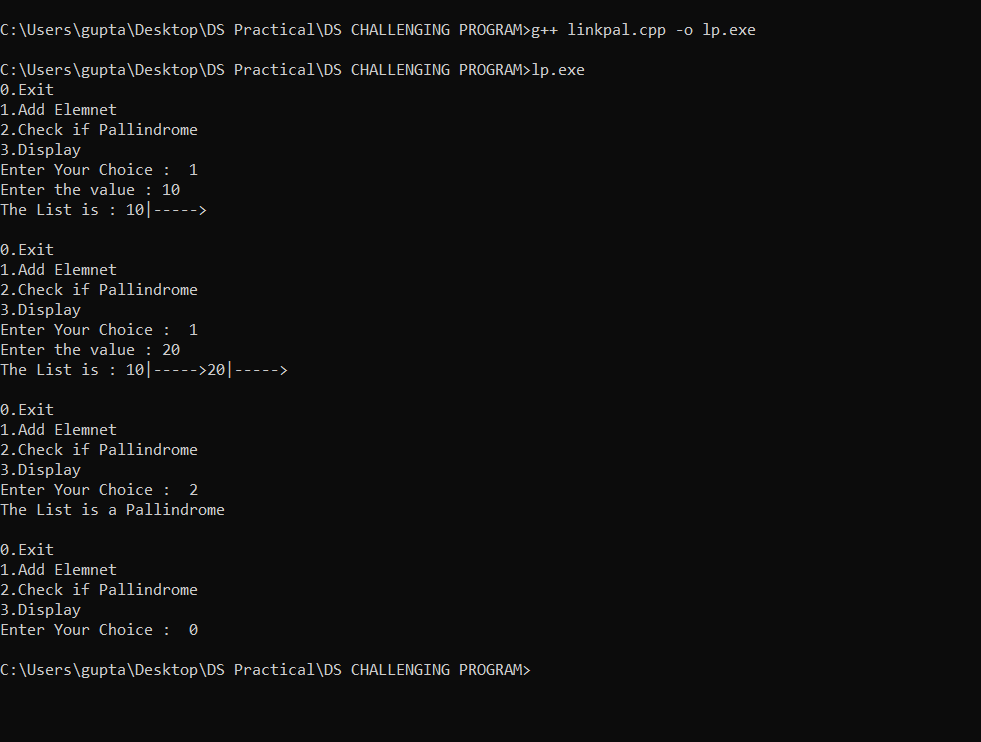
LinkPal s;

s.get();

return 0;

}

**Output :**

****

**Implement a program of double stack using single array**

**#include<iostream>**

**#include<stdlib.h>**

**using namespace std;**

**class SingleArrayDoubleStack**

**{**

**public:**

**int choice,value,ltop,rtop,size,rElement,count;**

**int arr[100];**

**SingleArrayDoubleStack()**

**{**

**ltop=-1;**

**cout<<"Enter Size of Array (Less than 100) : ";**

**cin>>size;**

**cout<<endl;**

**for(int i=0;i<size;i++)**

**{**

**arr[i]=0;**

**}**

**rtop=size;**

**}**

**void get()**

**{**

**do**

**{**

**cout<<"0.Exit\n01.Push an Element ( Left Side )\n02.Pop an Element ( Left Side )\n03.Push an Element ( Right Side )\n04.Pop an Element ( Right Side )\n05.Display\n";**

**cout<<"Enter Your Choice : "<<" ";**

**cin>>choice;**

**switch(choice)**

**{**

**case 0:**

**break;**

**case 1:**

**lpush();**

**break;**

**case 2:**

**lpop();**

**break;**

**case 3:**

**rpush();**

**break;**

**case 4:**

**rpop();**

**break;**

**case 5:**

**display();**

**break;**

**default:**

**cout<<"invalid input"<<endl<<endl;**

**}**

**}while(choice!=0);**

**}**

**bool isEmptyr()**

**{**

**if(rtop==size)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool isFullr()**

**{**

**if(arr[rtop-1] != 0)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool isEmptyl()**

**{**

**if(ltop==-1)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool isFulll()**

**{**

**if(arr[ltop+1] != 0)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**void lpush()**

**{**

**cout<<"Enter value : ";**

**cin>>value;**

**cout<<endl;**

**if(isFulll())**

**{**

**cout<<"Sorry OverFlowwwww "<<endl;**

**}**

**else**

**{**

**ltop=ltop+1;**

**arr[ltop]=value;**

**}**

**display();**

**cout<<endl;**

**}**

**void lpop()**

**{**

**cout<<endl;**

**if(isEmptyl())**

**{**

**cout<<"Sorry UnderFlow "<<endl;**

**}**

**else**

**{**

**rElement=arr[ltop];**

**arr[ltop]=0;**

**cout<<"The removed Element : "<<rElement<<" from Position : "<<ltop<<endl;**

**ltop=ltop-1;**

**}**

**display();**

**cout<<endl;**

**}**

**void rpush()**

**{**

**cout<<"Enter value : ";**

**cin>>value;**

**cout<<endl;**

**if(isFullr())**

**{**

**cout<<rtop<<" ";**

**cout<<"Sorry OverFlow "<<endl;**

**}**

**else**

**{**

**rtop=rtop-1;**

**arr[rtop]=value;**

**}**

**display();**

**cout<<endl;**

**}**

**void rpop()**

**{**

**cout<<endl;**

**if(isEmptyr())**

**{**

**cout<<"Sorry UnderFlow "<<endl;**

**}**

**else**

**{**

**rElement=arr[rtop];**

**arr[rtop]=0;**

**cout<<"The removed Element : "<<rElement<<" from Position : "<<rtop<<endl;**

**rtop=rtop+1;**

**}**

**display();**

**cout<<endl;**

**}**

**void display()**

**{**

**cout<<"The Elements in Stack are :"<<endl;**

**for(int i=0;i<size;i++)**

**{**

**cout<<"Position : "<<i<<" value : "<<arr[i]<<endl;**

**}**

**}**

**};**

**int main()**

**{**

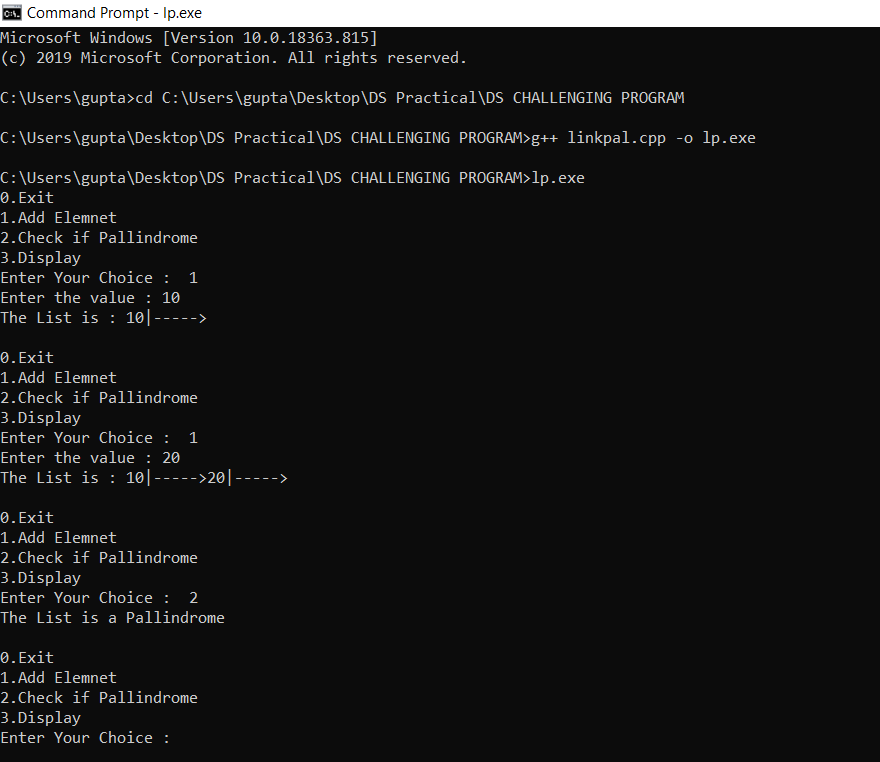
**SingleArrayDoubleStack d;**

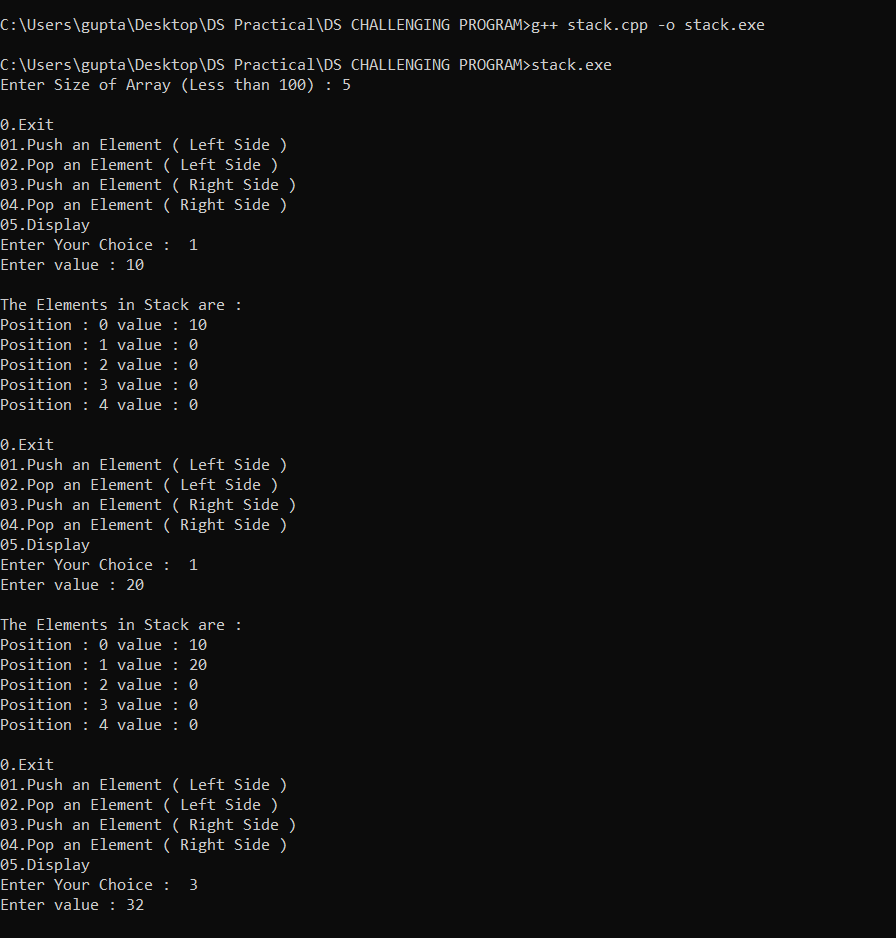
**d.get();**

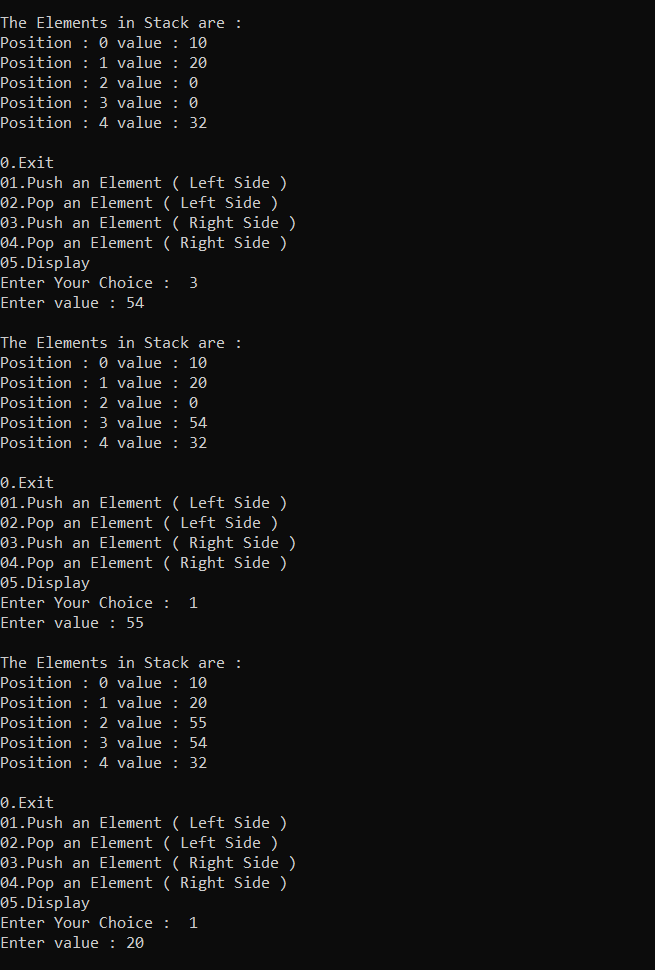
**return 0;**

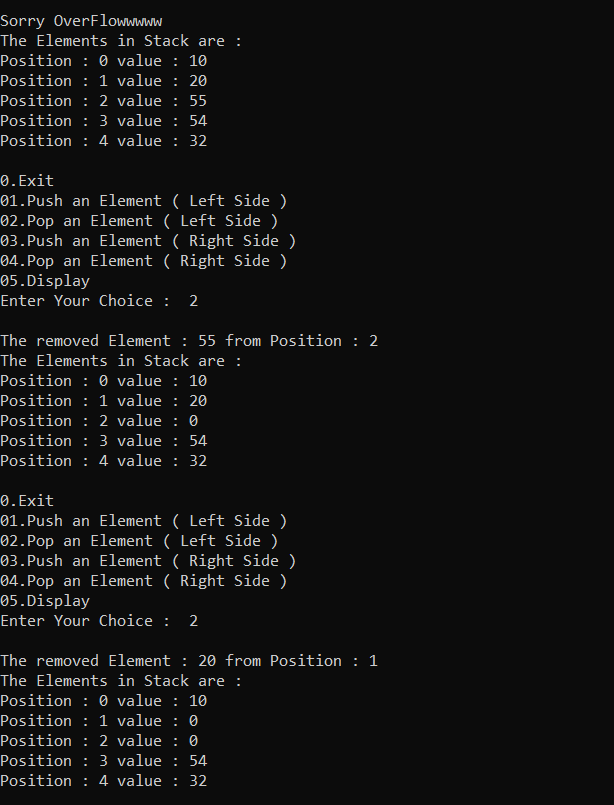
**}**

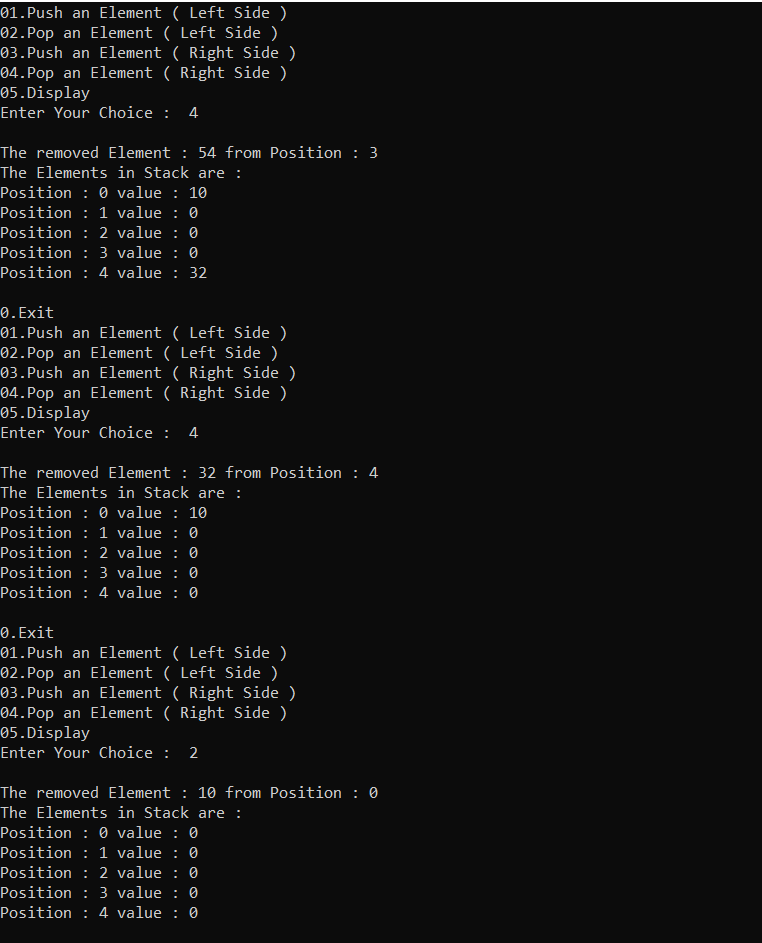
**Output :**

****

****

****

****

****

**Implement a program of Priority Queue**

**#include<iostream>**

**#include<stdlib.h>**

**using namespace std;**

**class PriorityQueue**

**{**

**public:**

**int choice,value,rear,front,rElement,count,size;**

**int arr[100],priority[100];**

**PriorityQueue()**

**{**

**front=0;**

**rear=-1;**

**cout<<"Enter Size of Array (Less than 100) : ";**

**cin>>size;**

**cout<<endl;**

**for(int i=0;i<size;i++)**

**{**

**arr[i]=0;**

**}**

**for(int i=0;i<size;i++)**

**{**

**priority[i]=111;**

**}**

**}**

**void get()**

**{**

**do**

**{**

**cout<<"0.Exit\n01.Enter an Element\n02 Delete an Element\n03.Display\n";**

**cout<<"Enter Your Choice : "<<" ";**

**cin>>choice;**

**switch(choice)**

**{**

**case 0:**

**break;**

**case 1:**

**insert();**

**break;**

**case 2:**

**delete\_ele();**

**break;**

**case 3:**

**display();**

**break;**

**default:**

**cout<<"invalid input"<<endl<<endl;**

**}**

**}while(choice!=0);**

**}**

**bool isEmpty()**

**{**

**if(front>rear)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool isFulll()**

**{**

**if(rear>=size-1)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**void insert()**

**{**

**cout<<"Enter value : ";**

**cin>>value;**

**cout<<endl;**

**if(isFulll())**

**{**

**cout<<"Sorry OverFlowwwww "<<endl;**

**}**

**else**

**{**

**rear=rear+1;**

**arr[rear]=value;**

**cout<<"Enter priority : ";**

**cin>>priority[rear];**

**cout<<endl;**

**}**

**display();**

**cout<<endl;**

**}**

**void delete\_ele()**

**{**

**int lowest\_priority,index\_lowest\_priority;**

**cout<<endl;**

**if(isEmpty())**

**{**

**cout<<"Sorry UnderFlow "<<endl;**

**}**

**else**

**{**

**lowest\_priority=priority[0];**

**for(int z=0;z<size;z++)**

**{**

**if(lowest\_priority >= priority[z])**

**{**

**lowest\_priority=priority[z];**

**index\_lowest\_priority=z;**

**}**

**}**

**rElement=arr[index\_lowest\_priority];**

**arr[index\_lowest\_priority]=0;**

**priority[index\_lowest\_priority]=111;**

**cout<<"The removed Element : "<<rElement<<" from Position : "<<index\_lowest\_priority<<endl;**

**front=front+1;**

**}**

**display();**

**cout<<endl;**

**}**

**void display()**

**{**

**cout<<"The Elements in Queue are :"<<endl;**

**for(int i=0;i<size;i++)**

**{**

**cout<<"Position : "<<i<<" value : "<<arr[i]<<" priority : "<<priority[i]<<endl;**

**}**

**}**

**};**

**int main()**

**{**

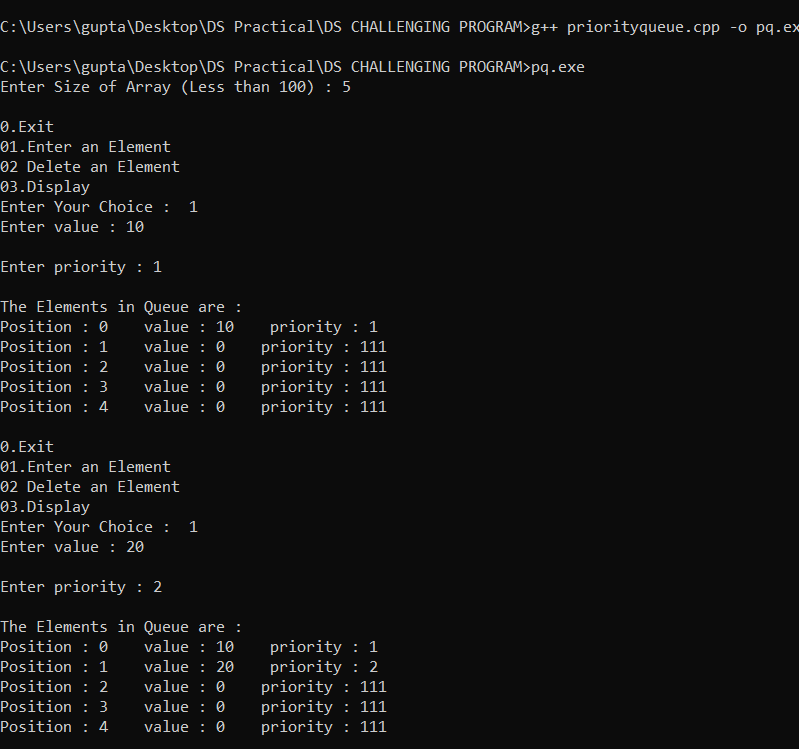
**PriorityQueue d;**

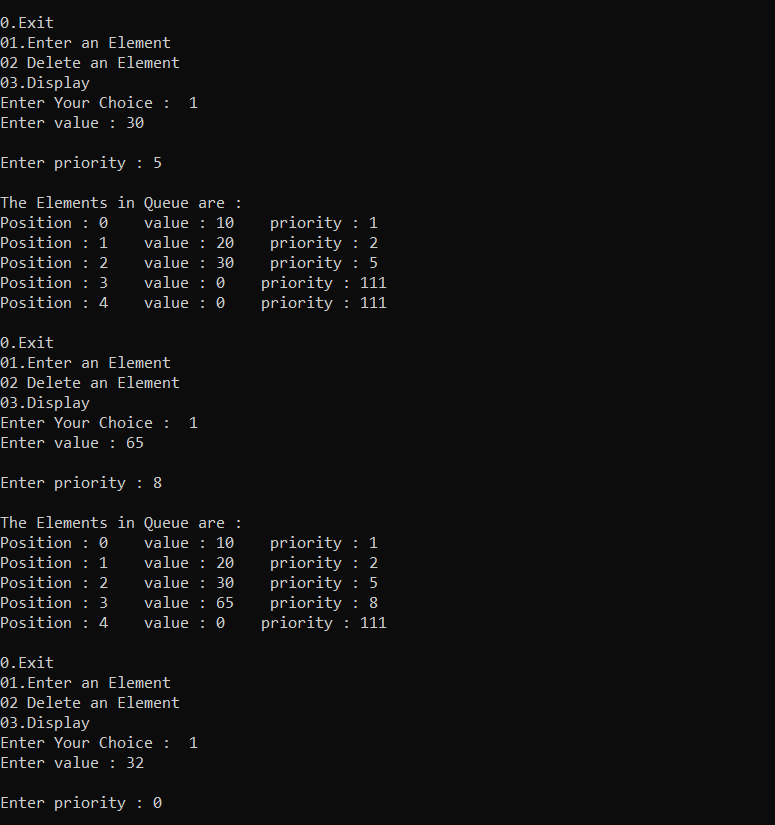
**d.get();**

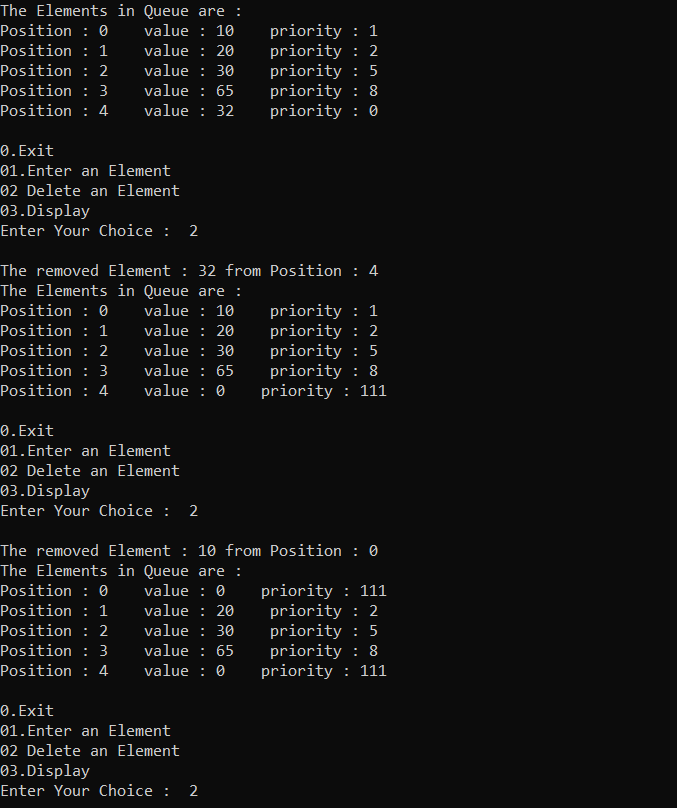
**return 0;**

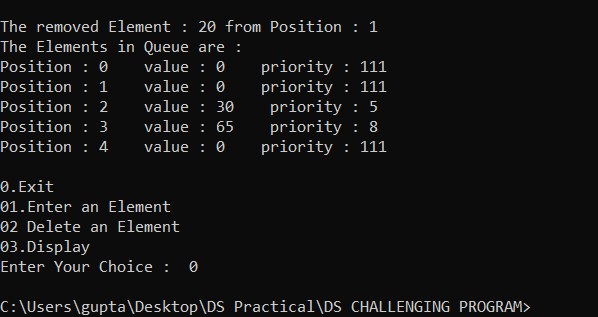
**}**

**Output :**

****

****

****

****