**4.1 Queue\_17.cpp**

**#include<iostream>**

**#include<stdlib.h>**

**using namespace std;**

**class Queue**

**{**

**public:**

**int choice,value,fr,rear,size,rElement;**

**int arr[100];**

**Queue()**

**{**

**rear=fr=-1;**

**for(int i=0;i<100;i++)**

**{**

**arr[i]=0;**

**}**

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

**cin>>size;**

**cout<<endl;**

**}**

**void get()**

**{**

**do**

**{**

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

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

**cin>>choice;**

**switch(choice)**

**{**

**case 0:**

**break;**

**case 1:**

**push();**

**break;**

**case 2:**

**pop();**

**break;**

**case 3:**

**display();**

**break;**

**default:**

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

**}**

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

**}**

**bool isEmpty()**

**{**

**if(rear == -1 && fr == -1)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool isFull()**

**{**

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

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**void push()**

**{**

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

**cin>>value;**

**cout<<endl;**

**if(isFull())**

**{**

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

**}**

**else if (rear == -1 and fr == -1)**

**{**

**rear=0;**

**fr=0;**

**arr[rear]=value;**

**}**

**else**

**{**

**rear=rear+1;**

**arr[rear]=value;**

**}**

**display();**

**cout<<endl;**

**}**

**void pop()**

**{**

**cout<<endl;**

**if(isEmpty())**

**{**

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

**}**

**else if (fr == rear)**

**{**

**rElement=arr[fr];**

**arr[fr]=0;**

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

**fr=-1;**

**rear=-1;**

**}**

**else**

**{**

**rElement=arr[fr];**

**arr[fr]=0;**

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

**fr=fr+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]<<endl;**

**}**

**}**

**};**

**int main()**

**{**

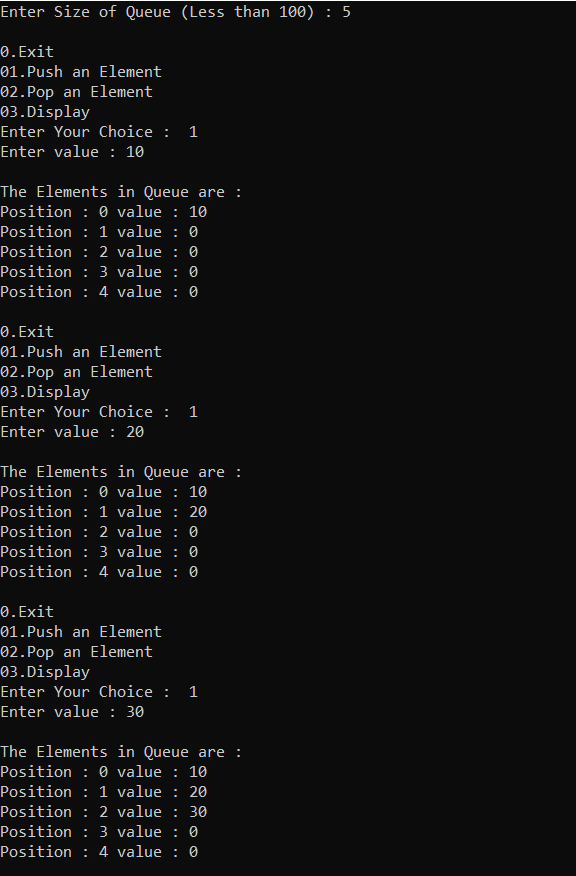
**Queue d;**

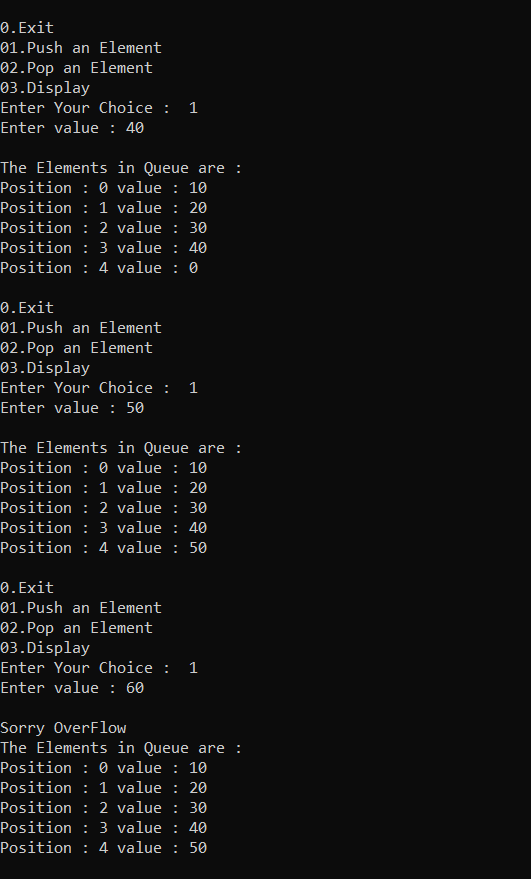
**d.get();**

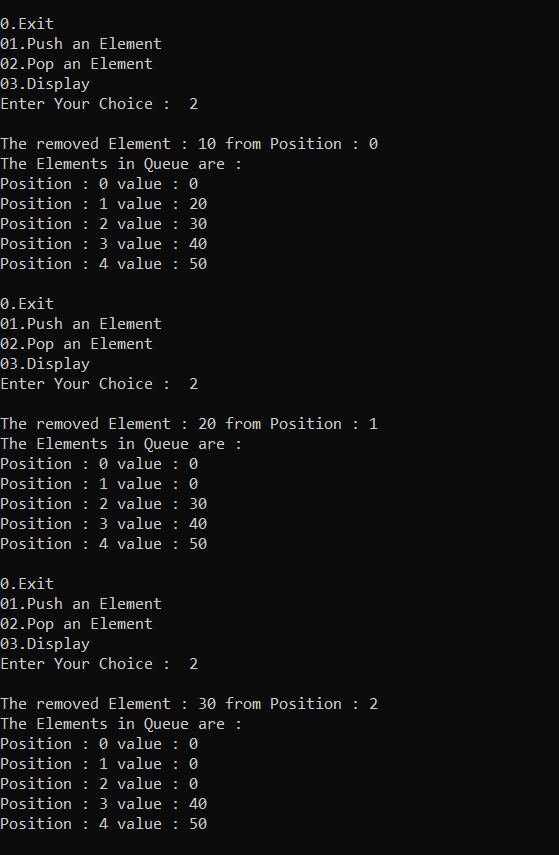
**return 0;**

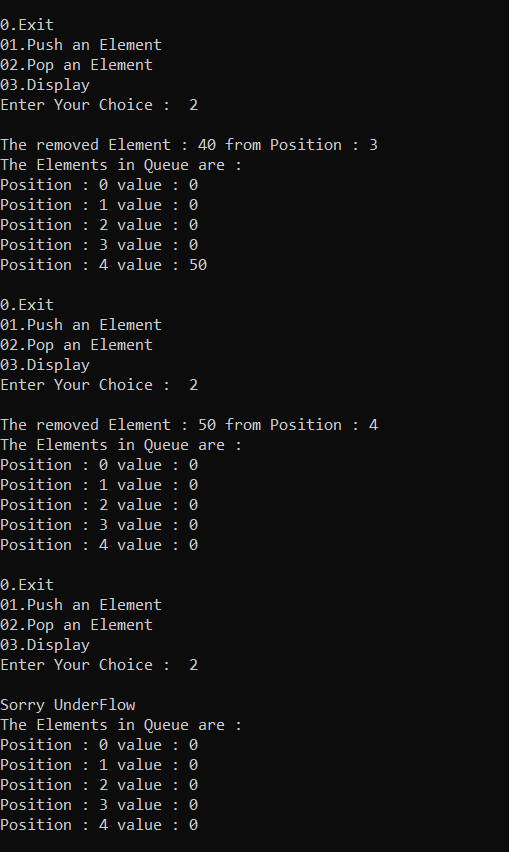
**}**

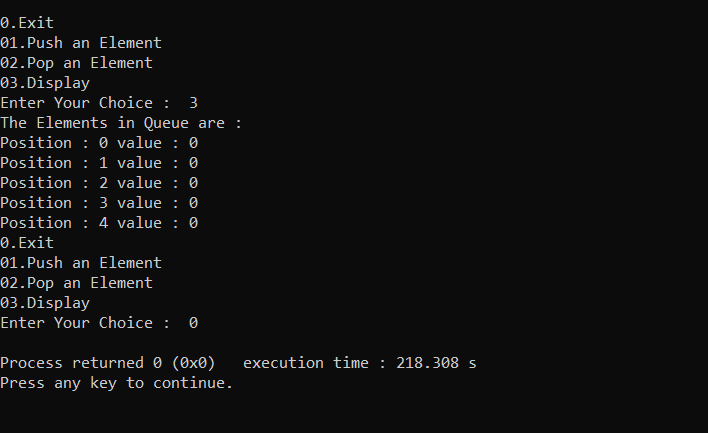
**Output :**

****

****

****

****

****

**4.2 Doubly\_Queue\_17.cpp**

**#include<iostream>**

**#include<stdlib.h>**

**using namespace std;**

**class DoublyQueue**

**{**

**public:**

**int choice,value,lfront,lrear,rrear,rfront,size,rElement,count;**

**int arr[100];**

**DoublyQueue()**

**{**

**lfront=lrear=-1;**

**for(int i=0;i<100;i++)**

**{**

**arr[i]=0;**

**}**

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

**cin>>size;**

**cout<<endl;**

**rrear=rfront=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(rrear == size && rfront == size)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool isFullr()**

**{**

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

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool isEmptyl()**

**{**

**if(lrear == -1 && lfront == -1)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool isFulll()**

**{**

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

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**void lpush()**

**{**

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

**cin>>value;**

**cout<<endl;**

**if(isFulll())**

**{**

**cout<<lrear<<" ";**

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

**}**

**else if (lrear == -1 and lfront == -1)**

**{**

**cout<<lrear<<" ";**

**lrear=0;**

**lfront=0;**

**arr[lrear]=value;**

**}**

**else**

**{**

**cout<<lrear<<" ";**

**lrear=lrear+1;**

**arr[lrear]=value;**

**}**

**display();**

**cout<<endl;**

**}**

**void lpop()**

**{**

**cout<<endl;**

**if(isEmptyl())**

**{**

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

**}**

**else if (lfront <= lrear)**

**{**

**rElement=arr[lfront];**

**arr[lfront]=0;**

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

**lfront=lfront+1;**

**}**

**else**

**{**

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

**}**

**display();**

**cout<<endl;**

**}**

**void rpush()**

**{**

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

**cin>>value;**

**cout<<endl;**

**if(isFullr())**

**{**

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

**}**

**else if (rrear == size and rfront == size)**

**{**

**rrear=size-1;**

**rfront=size-1;**

**arr[rrear]=value;**

**}**

**else**

**{**

**rrear=rrear-1;**

**arr[rrear]=value;**

**}**

**display();**

**cout<<endl;**

**}**

**void rpop()**

**{**

**cout<<endl;**

**if(isEmptyr())**

**{**

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

**}**

**else if (rrear <= rfront)**

**{**

**rElement=arr[rfront];**

**arr[rfront]=0;**

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

**rfront=rfront-1;**

**}**

**else**

**{**

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

**}**

**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]<<endl;**

**}**

**}**

**};**

**int main()**

**{**

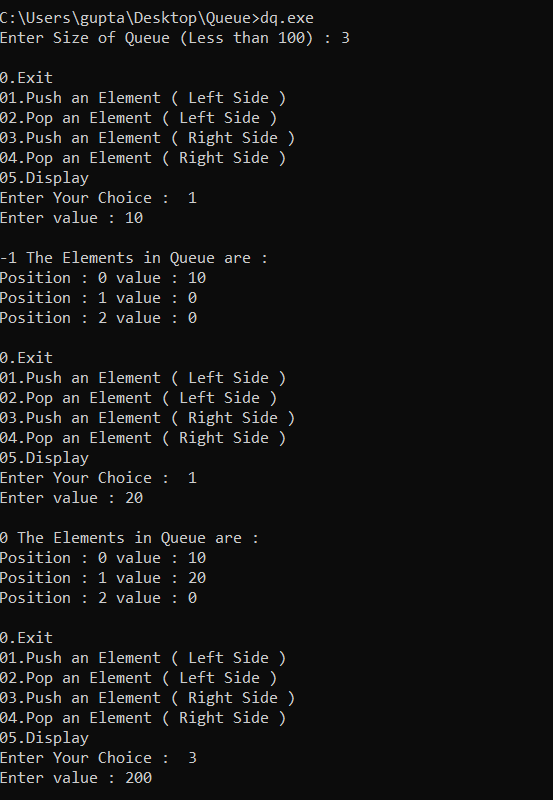
**DoublyQueue d;**

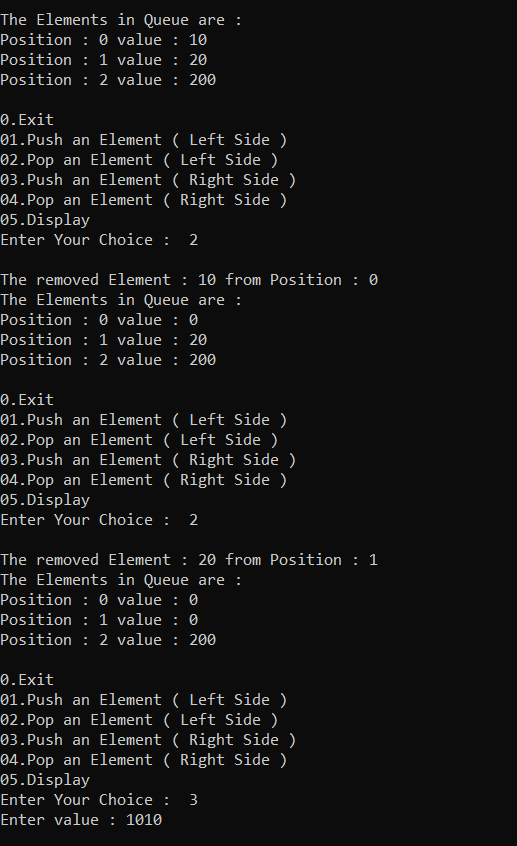
**d.get();**

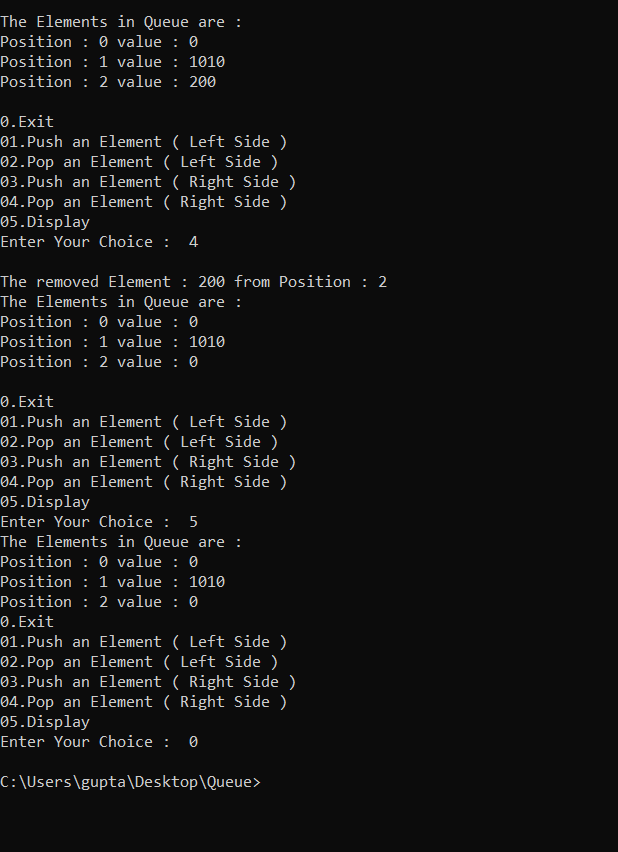
**return 0;**

**}**

**Output :**

****

****

****

**4.3 Circular\_Queue\_17.cpp**

**#include<iostream>**

**#include<stdlib.h>**

**using namespace std;**

**class CircularQueue**

**{**

**public:**

**int choice,value,fr,rear,size,rElement;**

**int arr[100];**

**CircularQueue()**

**{**

**rear=fr=-1;**

**for(int i=0;i<100;i++)**

**{**

**arr[i]=0;**

**}**

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

**cin>>size;**

**cout<<endl;**

**}**

**void get()**

**{**

**do**

**{**

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

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

**cin>>choice;**

**switch(choice)**

**{**

**case 0:**

**break;**

**case 1:**

**push();**

**break;**

**case 2:**

**pop();**

**break;**

**case 3:**

**display();**

**break;**

**default:**

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

**}**

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

**}**

**bool isEmpty()**

**{**

**if(rear == -1 && fr == -1)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool isFull()**

**{**

**if((rear+1)%size == fr)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**void push()**

**{**

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

**cin>>value;**

**cout<<endl;**

**if(isFull())**

**{**

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

**}**

**else if (rear == -1 and fr == -1)**

**{**

**rear=0;**

**fr=0;**

**arr[rear]=value;**

**}**

**else**

**{**

**rear=(rear+1)%size;**

**arr[rear]=value;**

**}**

**display();**

**cout<<endl;**

**}**

**void pop()**

**{**

**cout<<endl;**

**if(isEmpty())**

**{**

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

**}**

**else if (fr == rear)**

**{**

**rElement=arr[fr];**

**arr[fr]=0;**

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

**fr=-1;**

**rear=-1;**

**}**

**else**

**{**

**rElement=arr[fr];**

**arr[fr]=0;**

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

**fr=(fr+1)%size;**

**}**

**display();**

**cout<<endl;**

**}**

**void display()**

**{**

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

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

**{**

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

**}**

**}**

**};**

**int main()**

**{**

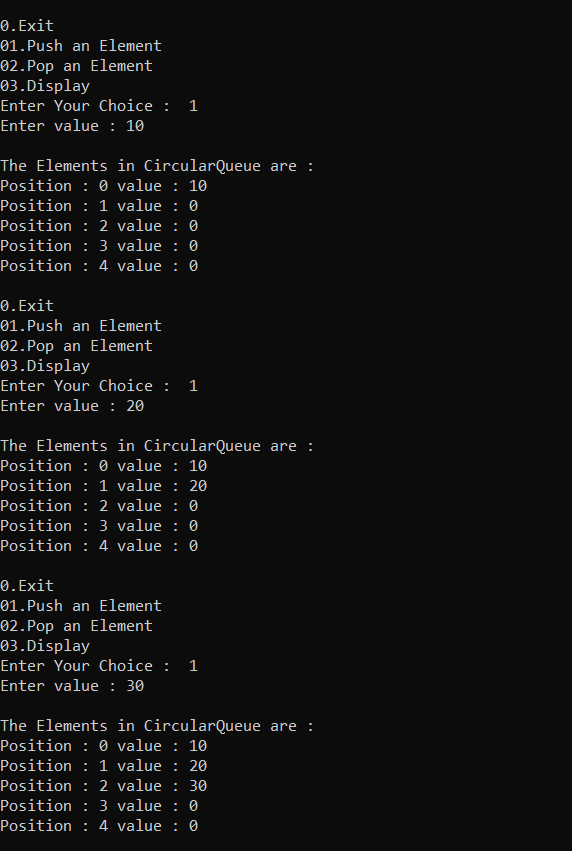
**CircularQueue d;**

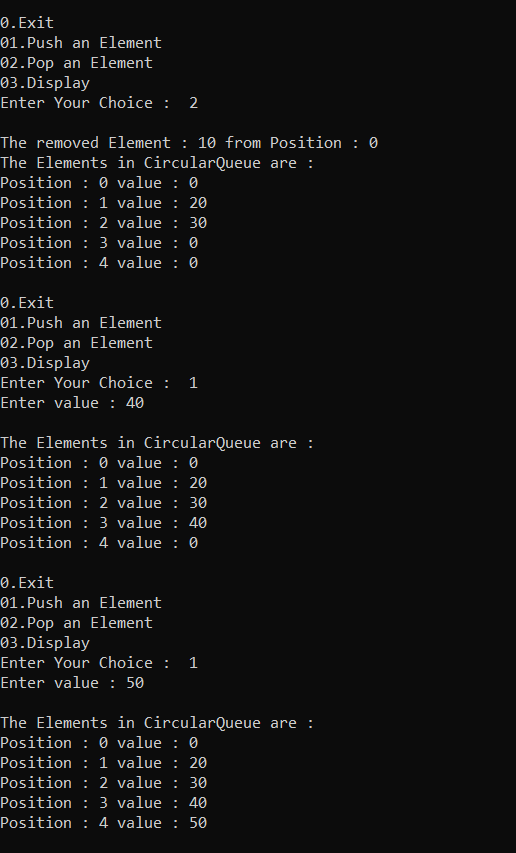
**d.get();**

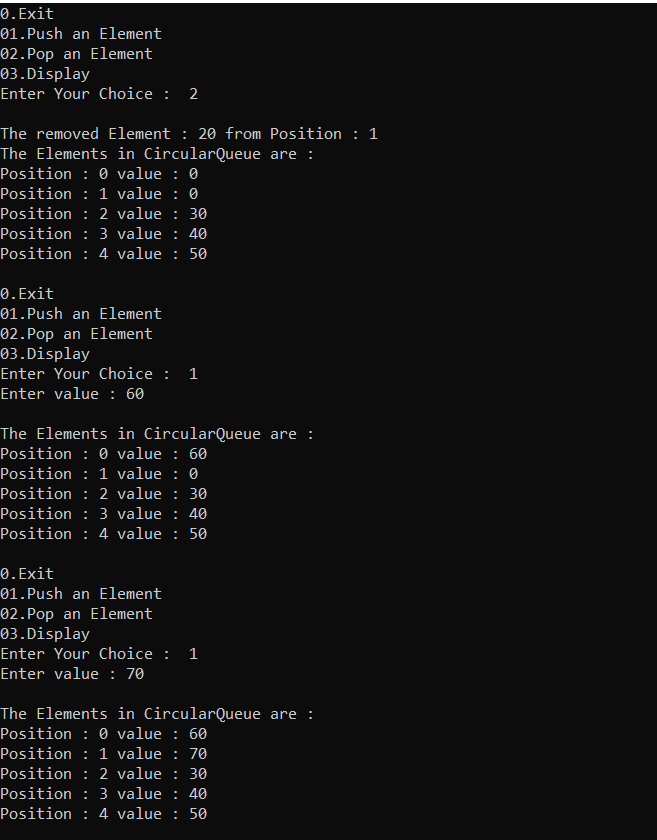
**return 0;**

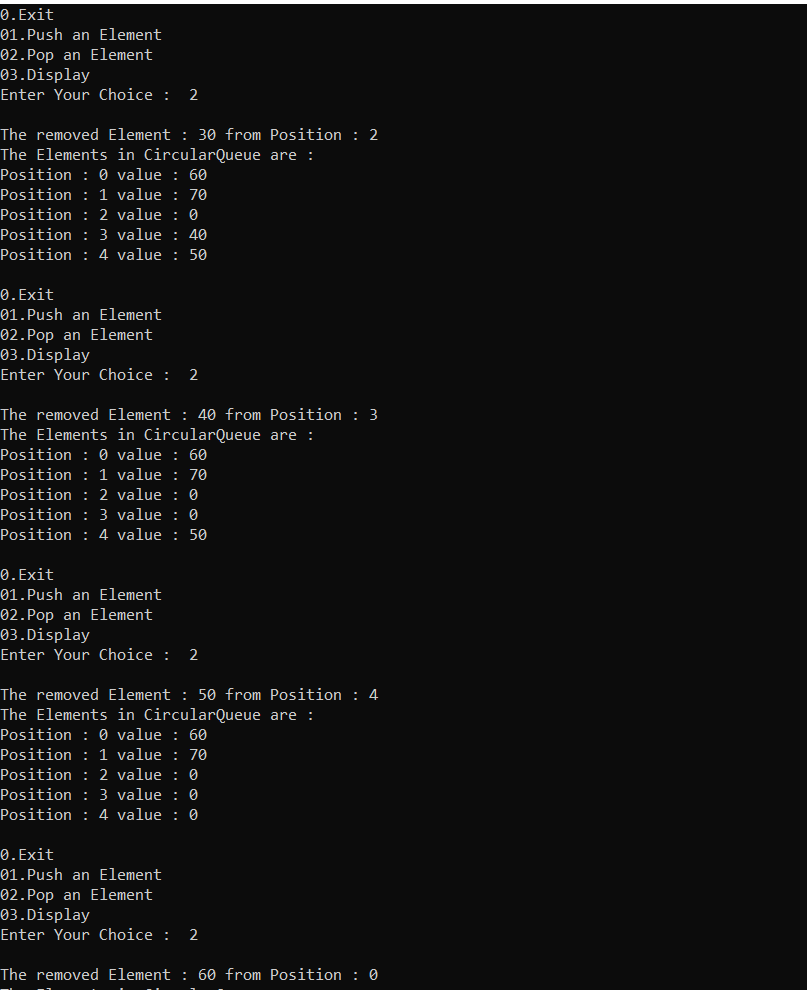
**}**

**Output :**

****

****

****

****

**4.4 Queue Using LinkedList**

**#include<iostream>**

**#include<stdlib.h>**

**using namespace std;**

**struct node**

**{**

**int data;**

**struct node \*next;**

**}**

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

**class QueueLink**

**{**

**public:**

**int choice,value;**

**void get()**

**{**

**do**

**{**

**cout<<"0.Exit\n1.Push\n2.Pop\n3.display\n";**

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

**cin>>choice;**

**switch(choice)**

**{**

**case 0:**

**break;**

**case 1:**

**push();**

**break;**

**case 2:**

**pop();**

**break;**

**case 11:**

**display();**

**break;**

**default:**

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

**}**

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

**}**

**void push()**

**{**

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

**cin>>value;**

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

**p->data=value;**

**if(list == NULL)**

**{**

**p->next=NULL;**

**list=p;**

**rear=p;**

**front=p;**

**}**

**else**

**{**

**rear->next=p;**

**p->next=NULL;**

**rear=rear->next;**

**}**

**display();**

**}**

**void pop()**

**{**

**cout<<"Delete Fisrt element "<<endl;**

**if(front == NULL && rear == NULL)**

**{**

**cout<<"Empty List"<<endl<<endl;**

**}**

**else if (rear == front)**

**{**

**p=front;**

**list=list->next;**

**front=front->next;**

**delete p;**

**}**

**else**

**{**

**p=front;**

**list=list->next;**

**front=front->next;**

**delete p;**

**}**

**display();**

**}**

**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()**

**{**

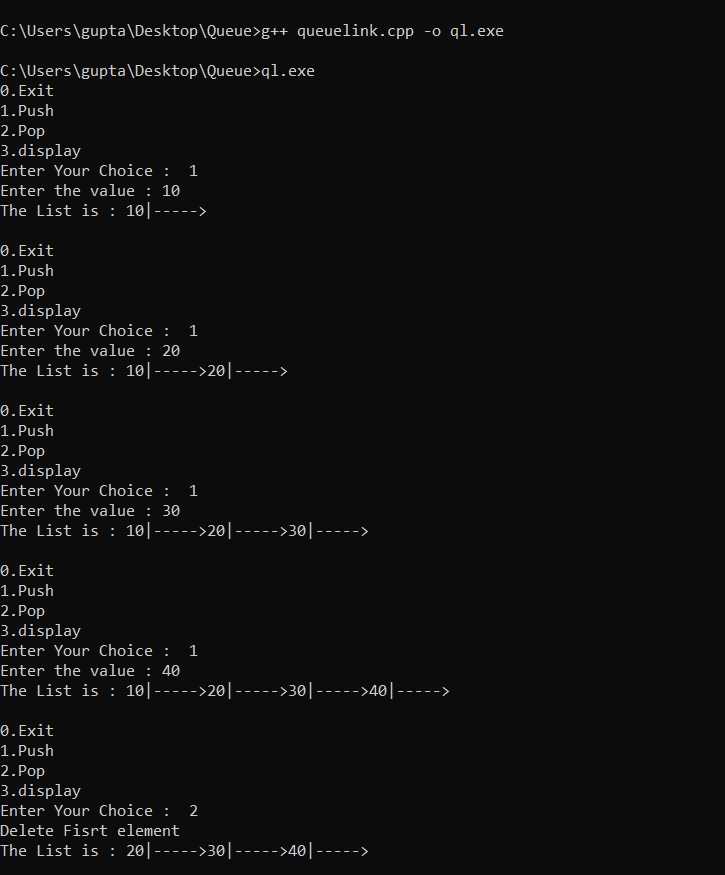
**QueueLink s;**

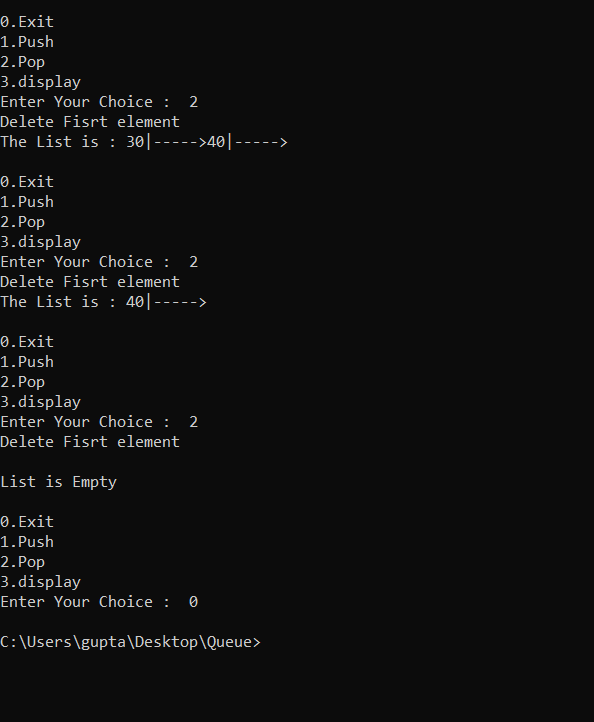
**s.get();**

**return 0;**

**}**

**Output :**

****

****