**Write a program for Circular implementation of a Queue.**

#include<iostream.h>

#include<conio.h>

struct Node

{

int data;

struct Node\*next;

};

Node\*front=NULL;

Node\*rear=NULL;

void enqueue(int val)

{

if(front==NULL||rear==NULL)

{

Node\*newNode;

newNode = new Node;

newNode->data=val;

newNode->next=NULL;

front=newNode;

rear=newNode;

}

else

{

Node\*newNode;

newNode=new Node;

newNode->data=val;

rear->next=newNode;

newNode->next=front;

rear=newNode;

}

}

void dequeue()

{

Node\*n;

n=front;

front=front->next;

delete(n);

}

void display()

{

Node\*ptr;

ptr=front;

do

{

cout<<ptr->data<<" ";

ptr=ptr->next;

}

while(ptr!=rear->next);

cout<<endl;

cout<<endl;

}

int main(void)

{

enqueue(10);

enqueue(20);

enqueue(30);

enqueue(40);

enqueue(50);

enqueue(60);

display();

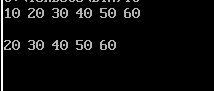
dequeue();

display();

return 0;

}

**OUTPUT :-**



**write a progran for array implementation of a Satck.**

#include<iostream.h>

#include<conio.h>

int stack[100], n=100, top=-1;

void push(int val)

{

if(top>=n-1)

cout<<"Stack Overflow"<<endl;

else

{

top++;

stack[top]=val;

}

}

void pop()

{

if(top<=-1)

cout<<"Stack Underflow"<<endl;

else

{

cout<<"The popped element is"<<stack[top]<<endl;

top--;

}

}

void display()

{

if(top>=0)

{

cout<<"Stack element are:";

for(int i=top;i>=0;i--)

cout<<stack[i]<<" ";

cout<<endl;

}

else

cout<<"Stack is empty";

}

int main()

{

int ch,val;

cout<<"1)Push in stack"<<endl;

cout<<"2)Pop from stack"<<endl;

cout<<"3)Display stack"<<endl;

cout<<"4)Exit"<<endl;

do

{

cout<<"Enter choice:";

cin>>ch;

switch(ch)

{

case 1:

{

cout<<"Enter value to be pushed:";

cin>>val;

push(val);

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

{

cout<<"Exit"<<endl;

break;

}

default:

{

cout<<"Invalid Choice"<<endl;

}

}

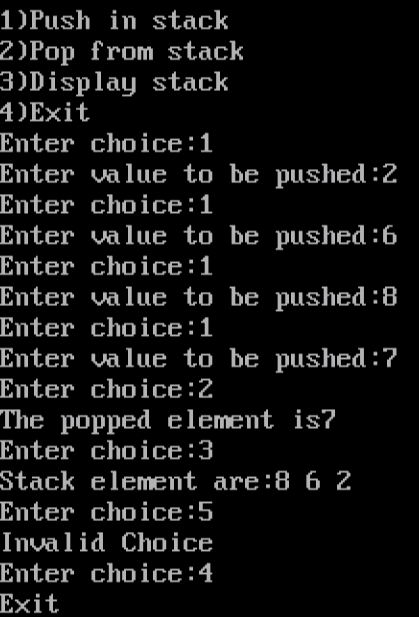
}

while(ch!=4);

return 0;

}

**OUTPUT** :-



**Write a program for link list implementation of a Stack.**

#include<iostream.h>

#include<conio.h>

struct node

{

int info;

struct node\*next;

};

class Queue

{

private:

node\*rear;

node\*front;

public:

Queue();

void enqueue();

void dequeue();

void display();

};

Queue::Queue()

{

rear=NULL;

front=NULL;

}

void Queue::enqueue()

{

int data;

node\*temp=new node;

cout<<"Enter the data to enqueue:";

cin>>data;

temp->next=NULL;

if(front==NULL)

{

front=temp;

}

else

{

rear->next=temp;

}

rear=temp;

}

void Queue::dequeue()

{

node\*temp=new node;

if(front==NULL)

{

cout<<"\nQueue is Empty\n";

}

else

{

temp=front;

front=front->next;

cout<<"The data Dequeue is"<<temp->info;

delete temp;

}}

void Queue::display()

{

node\*p=new node;

p=front;

if(front==NULL)

{

cout<<"\n nothing to Display\n";

}

else

{

while(p!=NULL)

{

cout<<endl<<p->info;

p=p->next;

}

}

}

int main()

{

Queue queue;

int choice,true;

while(true)

{

cout<<"\n 1.Emqueue\n 2.Dequeue \n 3.Display \n 4.Quit";

cout<<"\n Enter your choice:";

cin>>choice;

switch(choice)

{

case 1:

queue.enqueue();

break;

case 2:

queue.dequeue();

break;

case 3:

queue.display();

break;

case 4:

defult:

cout<<"\n Invalid Input. Try again!\n";

break;

}

}

return 0;

}

**OUTPUT :-**