CODES FOR STACK AND QUEUE (5 CODES)

STACK

Stack is LIFO structure and physically it can be implemented as array(Static Stack) or as a Linked List(Dynamic Stack). An Insertion in Stack is called PUSH and Deletion is called POP. Insertion and deletion always occurs at one end i.e. top.

STACK AS AN ARRAY

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
using namespace std;
const int size=10;
class stack
{
        int s[size];
        int top;
        public:
        stack()
        {
        top=-1;
        }
        void push();
        void pop();
        void disp();
};
void stack::push()
{
        if(top==size-1)
        { cout<<"OVERFLOW error"; getch(); return; }
        else
        {++top;
                cin>>s[top];  }
}

void stack::pop()
{
        if(top==-1)
        {
        cout<<"Underflow";getch(); return;}
        else
        {cout<<"\n\nElement deleted "<<s[top--];getch();}
}
void stack::disp()
```

```
{ for(int i=top;i>=0;i--) cout<<"\n Element "<<s[i]; getch();}
main()
{ stack s;int ch;
do
{  system("cls"); cout<<"\n 1. Push \n 2. Pop \n 3.Display \n 4.exit\n enterchoice";
cin>>ch;
switch(ch)
{ case 1: s.push(); break;
case 2:s.pop();break; case 3: s.disp();break; case 4: exit(1);} }while(ch!=4);
}
```

- In case of exit it will come out form the program, in case of return it will come out from the called function.
- In case of non member functions pass the arguments i.e.
  Void push(int a[],int &top, int ele)

DYNAMIC STACK (LINKED LIST IMPLMENTATION)

```
#include<conio.h>
#include<process.h>
#include<iostream>
using namespace std;
struct node{
int info;
node *next;
};
class stack
{
node *top;
public:
stack() { top=NULL;}
void push(); void pop(); void disp(); };
void stack::push()
{
//Creation of new node
node *temp=new node;
if(temp == NULL) { cout<<"OVERFLOW ERROR"; getch(); return; }
else { cout<<"Enter element ";cin>>temp->info; temp->next = NULL;}
//placing in stack
if(top==NULL) { top=temp; }
else { temp->next=top;     top=temp; } }


void stack::pop()
```

```
{
node *temp;
if(top==NULL)
{ cout<<"Underflow Error "; getch(); return; }
else
{temp=top; top=top->next; cout<<"Information Deleted "<<temp->info; getch();
delete temp; }
}
void stack::disp()
{
for(node *temp=top;temp!=NULL;temp=temp->next)
{cout<<"temp->info "<<temp->info<<"\n temp "<<temp<<"\n temp->next"
<<temp->next;getch();} }
int main()
{ stack s;int ch;
do
{ // system("cls");
cout<<"\n 1. Push \n 2. Pop \n 3.Display \n 4.exit\n enterchoice";
cin>>ch;
switch(ch)
{ case 1: s.push(); break;
case 2:s.pop();break; case 3: s.disp();break; case 4: return 0;} }while(ch!=4);
}
```

Applications of Stack

1. Reversing the string.
2. Converting Decimal to binary, Octal
3. Polish Notation (conversion pf Infix to Postfix/Prefix, Evaluation of Postfix/Prefix expression.

Note : In the question of Conversions , The Infix expression is given and postfix form is desired. Don't evaluate the expression until asked and be careful about the rules of conversion. In case of evaluation, postfix form is given.

Queue – Logically queue is a FIFO structure. And physically it can be implemented in terms of array as well as Linked List. Insertion takes place at rear end and deletion takes place at front end.

Static QUEUE

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
```

```
using namespace std;
const int size=10;
class queue
{
int q[10];
int f,r;
public:
queue()
{f=r=-1;}
void insert();void del(); void disp(); };
void queue::insert()
{
if(r==size-1)
{cout<<"Overflow error"; getch(); return; }
else
if(r==-1)
f=r=0;
else
r++;
cout<<"Enter element ";cin>>q[r]; }
void queue::del()
{    if(f==-1)
{cout<<"Underflow error"; getch(); return; }
else
{    cout<<"Value deleted "<<q[f]; getch();
if(f==r)
f=r=-1;
else
f++; } }
void queue::disp()
{for(int i=f;i<=r;i++)

cout<<q[i]<<" ";  getch();}
 main()
{ queue q;int ch;
do
{   cout<<"\n 1. Insert \n 2. Delete \n 3.Display \n 4.exit\n enterchoice";
cin>>ch;
switch(ch)
{ case 1: q.insert(); break;
case 2:q.del();break; case 3: q.disp();break; case 4: exit(1);} }while(ch!=4);
}
```

Circular Queue (Static – ARRAY IMPLEMENTATION)

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;
const int size = 5;
class queue{
        int ar[size];
        int front, rear,tot_elements;
        public:
        queue()
        {
        front = 0; rear = -1;
        tot_elements=0;
        }
        void ins(int ele);
        int del();
        void disp();
        };
void queue::ins(int ele)
{   if(tot_elements<size)
{
        rear=(rear+1)%size;
        tot_elements +=1;
        ar[rear]=ele;    }
else
{   cout<<"OVERFLOW.......!!!";   getch(); exit(0);    }  }
int queue::del()
{
int val;
if(tot_elements>0)
{
        val=ar[front];
        front=(front+1)%size;
        tot_elements -= 1;  }
else
{  cout<<"UNDERFLOW.......!!!"; getch();  exit(1); }
return val;  }
void queue::disp()
{
        for(int i=1,temp=front ;  i<=tot_elements ;  i++)
        {       cout<<ar[temp];
        temp=(temp+1)%size;
        getch();      }       }
int main()
{ queue q;
int ch,val;
do{
cout<<"\n\n1. Add element \n2. Delete element \n3.Display \n0.Exit\n ENter
choice";
cin>>ch;
switch(ch)
{       case 1: cout<<"enter element";
                        cin>>val;
                        q.ins(val);break;
        case 2: cout<<q.del()<<" Deleted ";break;
        case 3: q.disp(); break;
        case 0: return 0;
        default : cout<<"Invalid"; getch();  }
}while(ch!=0);   }
```

<u>Dynamic Queue(LINKED LIST IMPEMENTATION)</u>

```
#include<conio.h>
#include<process.h>

#include<iostream>
using namespace std;
struct node{
int info;
```

```
node *next;
};
class queue
{
node *f,*r;
public:
queue()
 { f=NULL;
 r=NULL;}
void ins(); void del(); void disp(); };
void queue::ins()
{
//Creation of new node
node *temp;  temp=new node;
if(temp == NULL) { cout<<"Queue is full"; getch(); return; }
else { cout<<"Enter element ";cin>>temp->info; temp->next = NULL;}
//placing in queue
if(r==NULL) { f=r=temp; }
else
 { r->next=temp;     r=temp } }
void queue::del()
{
node *temp;
if(f==NULL)
{ cout<<"Queue is empty "; getch(); return; }
else
{temp=f;
f=f->next; cout<<"Information Deleted "<<temp->info; getch();
delete temp; }
}
void queue::disp()
{
for(node *temp=f;temp!=NULL;temp=temp->next)
{cout<<"temp->info "<<temp->info<<"\n temp "<<temp<<"\n temp->next"<<temp->next;getch();}
}
 int main()
{
        queue s;int ch;
do
{   cout<<"\n 1. Insert \n 2. Delete \n 3.Display \n 4.exit\n enterchoice";
cin>>ch;
switch(ch)
{ case 1: s.ins(); break;
case 2:s.del();break; case 3: s.disp();break; case 4: return 0;}
}while(ch!=4);
}
```