ASSIGNMENT NO:1 **DATE:14/07/2015**

PROGRAM TITLE:Write a Program to implement a Stack or Queue using 1D array applying the concept of class.

THEORY: Stack and Queue are Linear Data Structures that can be implemented using an array or a Linked list. In this program we implement it using an array and also the concept of class.

A class is syntactically similar to a structure; but unlike a structure, a class can hold both data members(i.e. characteristics)as well as member functions(i.e. behaviour)that are called methods.

The stack has only one pointer top that points to the top of the stack. Stack follows the LIFO(Last In First Out) logic.
The simple queue has 2 pointers front and rear. The insert operation occurs in the rear and the items are deleted from the front. Queue follows the FIFO(First In First Out) logic.

PROGRAM ALGORITHM:

Stack:

```
//'top' points to the top of the stack. 'max' is the maximum size of the
array 'stk', on which we implement the stack.
Algo_push(item)
{
     if(top=max)
          print overflow;
     else
     {
          top=top+1;
          stk[top]=item;
     }
}

Algo_pop(top)
{
     if (top=0)
          print underflow;
     else
     {
          print that the item popped is q[top];
          top=top-1;
     }

}
```

Queue:

```
//'front' and 'rear' are pointing to the beginning and the end of the
queue. 'max' is the maximum size of the array 'q', on which we implement
the queue.
Algo_insert(item)
{

     if(rear=max)
     {
          if(front is not zero)//Provision for Inchworm Effect
               shift the elements of the array so as to fill the empty
spaces.
          else
               print queue is full.
     }
     else
     {
          q[rear]=item;
          rear=rear+1;
     }
}

Algo_delete()
{
     if (front+1=rear)
          print that queue is empty;
     else
          increase front by one;
}
```

PROGRAM CODE:

```
/*C++ Program to implement a Stack or Queue using 1D array applying the
concept of class.*/
#include<iostream>
using namespace std;

/*Class Stack and its associated functions*/
class Stack
{
     int *stk,max,top;
     public:
          Stack(int);
          int push(int);
          int pop();
};

/*Parameterised constructor of class Stack*/
Stack::Stack(int n)
{
     stk=new int[n];
     max=n;
     top=-1;
}

/*Function to implement pushing an element into the Stack*/
int Stack::push(int x)
{
     if(top+1==max)
          cout<<"\n\tOverflow"<<endl;
     else
```

```
	{
		stk[++top]=x;
	}
	return 0;
}

/*Function to implement popping an element from Stack*/
int Stack::pop()
{
	if(top==-1)
		cout<<"\n\tUnderflow"<<endl;
	else
	{
		cout<<"\n\tItem popped is "<<stk[top--]<<endl;
	}
	return 0;
}

/*Class Queue and its associated functions*/
class Queue
{
	int *q,max,front,rear;
	public:
		Queue(int);
		int insert(int);
		int del();
		void display()
		{
			if(front+1==rear)
				cout<<"\n\tQueue is empty"<<endl;
			for(int i=front+1;i<rear;i++)
			{
				cout<<"\t"<<q[i]<<endl;
			}
		}
};

/*Parameterised constructor of class Queue*/
Queue::Queue(int n)
{
	q=new int[n];
	max=n;
	rear=0;
	front=-1;
}

/*Function to implement insertion into a Queue*/
int Queue::insert(int x)
{
	if(rear==max)
	{

		/*Correcting the Inchworm Effect*/
		if(front!=-1)
		{
```

```
                for(int i=0;i<rear;i++)
                     q[i]=q[i+front+1];
                rear=rear-(front+1);
                front=-1;
                q[rear++]=x;
           }
           else
                cout<<"\n\tQueue is full."<<endl;
      }
      else
           q[rear++]=x;
      return 0;
}

/*Function to implement deletion from a Queue*/
int Queue::del()
{
      if(front+1==rear)
           cout<<"\n\tQueue is empty.";
      else
           ++front;
      return 0;
}
int main()
{
      int ch=0,n;

      /*Asking the user whether to create a Stack or a Queue*/
      cout<<"\n\tCreation Menu::\n\t1.Stack\n\t2.Queue\n\tYour Choice::";
      cin>>ch;
      switch(ch)
      {
           case 1: {
                cout<<"\n\tEnter size of Stack:";
                cin>>n;

                /*Creating an object of class Stack*/
                Stack stk(n);
                do
                {

                     /*Menu for implementing Stack functions*/
                     cout<<"\n\tMenu for 
Stack::\n\t1.Push\n\t2.Pop\n\t3.Exit\n\tYour Choice::";
                     cin>>ch;
                     switch(ch)
                     {
                          case 1: cout<<"\n\tEnter the item to be 
pushed:";
                                cin>>n;
                                stk.push(n);
                                break;
                          case 2: stk.pop();
                                break;
                          case 3: cout<<"\n\tProgram Terminated.\n";
```

```
                                break;
                           default:cout<<"\n\tInvalid Choice.";
                      }
                 }
                 while(ch!=3);
                 }
                 break;
            case 2: {
                 cout<<"\n\tEnter size of Queue:";
                 cin>>n;

                 /*Creating an object of class Queue*/
                 Queue q(n);
                 do
                 {

                      /*Menu for implementing Queue functions*/
                      cout<<"\n\tMenu for
Queue::\n\t1.Insert\n\t2.Delete\n\t3.Display\n\t4.Exit\n\tYour Choice::";
                      cin>>ch;
                      switch(ch)
                      {
                           case 1: cout<<"\n\tEnter the item to be
inserted:";
                                cin>>n;
                                q.insert(n);
                                break;
                           case 2: q.del();
                                break;
                           case 3: q.display();
                                break;
                           case 4: cout<<"\n\tProgram Terminated.\n";
                                break;
                           default:cout<<"\n\tInvalid Choice.";
                      }
                 }
                 while(ch!=4);
                 }
                 break;
            default:cout<<"\n\tInvalid Choice.";
        }
        return 0;
}
```

OUTPUT:

Stack:

```
        Creation Menu::
        1.Stack
        2.Queue
        Your Choice::1

        Enter size of Stack:5

        Menu for Stack::
```

```
1.Push
2.Pop
3.Exit
Your Choice::1

Enter the item to be pushed:10

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::2

Item popped is 10

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::2

Underflow

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::1

Enter the item to be pushed:20

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::1

Enter the item to be pushed:30

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::1

Enter the item to be pushed:40

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::1

Enter the item to be pushed:50
```

```
Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::1

Enter the item to be pushed:60

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::1

Enter the item to be pushed:70

Overflow

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::2

Item popped is 60

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::2

Item popped is 50

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::2

Item popped is 40

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::2

Item popped is 30

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::2
```

```
Item popped is 20

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::2

Underflow

Menu for Stack::
1.Push
2.Pop
3.Exit
Your Choice::3

Program Terminated.
```

Queue:

```
Creation Menu::
1.Stack
2.Queue
Your Choice::2

Enter size of Queue:4

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::1

Enter the item to be inserted:10

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::3
10

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::2

Menu for Queue::
1.Insert
2.Delete
3.Display
```

```
4.Exit
Your Choice::2

Queue is empty.
Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::1

Enter the item to be inserted:10

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::1

Enter the item to be inserted:20

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::1

Enter the item to be inserted:30

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::1

Enter the item to be inserted:40

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::3
10
20
30
40

Menu for Queue::
1.Insert
2.Delete
3.Display
```

```
4.Exit
Your Choice::1

Enter the item to be inserted:50

Queue is full.

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::2

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::2

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::3
30
40

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::1

Enter the item to be inserted:50

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::3
30
40
50

Menu for Queue::
1.Insert
2.Delete
3.Display
4.Exit
Your Choice::4
```

```
    Program Terminated.
```

DISCUSSION:

Stack:

1.The complexity of push is O(1).
2.The complexity of pop is O(1).
3.Overflow occurs only when the top reaches the maximum size of the array.

Queue:

1.The complexity of insertion is O(1).
2.The complexity of deletion is O(1).
3.Overflow occurs only when the top reaches the maximum size of the array.

We also correct the Inchworm effect in this program, i.e. we make sure that there are no empty spaces left in the array (as is the case when front doesn't point to the first position) before declaring it as empty.

While declaring the objects of the 'Stack' or 'Queue' class within the switch case, we faced a problem that the scope of the object was not clearly known. To overcome this problem we put all the statements of that particular case within a scope.