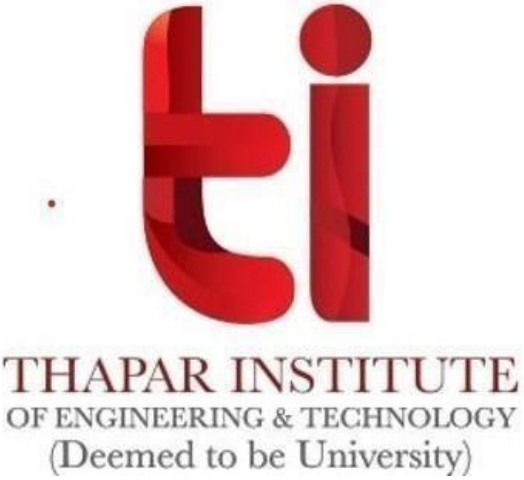
A Practical Activity Report For
Data Structures and Algorithms (UCS406)

Submitted By: **Vivek Arora**

101715178
(ENC 8)

Submitted To:

Dr. Sanjay Sharma



ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING &TECHNOLOGY, (DEEMED TO BEUNIVERSITY), PATIALA, PUNJAB

ASSIGNMENT 6

QUESTION 1:

Implement the operation of stack using array to store the stack data items; Stack change in size therefore you need to store the maximum number of data items the stack can hold (max_size) and the array index off the top most data item in the stack (top)along with the stack data items themselves base your implementation on the following declarations from the file stackarr.h. Implement the whole program using template class also.

//stackarr.h program

```
class Stack
{
   public:
   int *arr;
   int max_size;
   int top;
   Stack();
   void push(int );
   int pop();
   bool isfull();
   bool isempty();
};

Stack::Stack()
{
  max_size=5;
  arr=new int[max_size];
  top=-1;
}

void Stack::push(int data)
{
  ++top;
  arr[top]=data;
  cout<<arr[top]<<endl;
}

int Stack::pop()
{
return arr[top--];
}
```

```
bool Stack::isfull()
{
   return top==max_size-1;
}

bool Stack::isempty()
{
    return top==-1;
}

#include<iostream>
using namespace std;
#include"stackarr.h"
int main()
{
  int i;
  Stack S1;
  S1.push(12);
  S1.push(22);
  S1.push(32);
  S1.push(42);
  cout<<"the no popped out of stack is "<<S1.pop()<<endl;
}
```

USING TEMPALTE CLASS

```
//stacktemp program
template <class T>
class Stack
{
  public:
  T *arr;
  int max_size;
  int top;
  Stack();
  void push(T);
  T pop();
   bool isfull();
   bool isempty();
};

template <class T>
Stack<T>::Stack()
{max_size=5;
```

```
arr=new T[max_size];
top=-1;
}

template <class T>
void Stack<T>::push(T data)
{
  ++top;
  arr[top]=data;
  cout<<arr[top]<<endl;
}

template <class T>
T Stack<T>::pop()
{
return arr[top--];
}

template <class T>
bool Stack<T>::isfull()
{
  return top==max_size-1;
}

template <class T>
bool Stack<T>::isempty()
   {
         return top==-1;
   }

#include<iostream>
using namespace std;
#include"stacktemp.h"

int main()
{
 int i;
 Stack<int> S1;
 Stack<char*> S2;
 S1.push(12);
 S1.push(22);
 S1.push(32);
 S1.push(42);
```

```
 cout<<"the no popped element out of stack is "<<S1.pop()<<endl;
 S2.push("first");
 S2.push("second");
 S2.push("third");
 S2.push("last");
 cout<<"the string popped out of stack is "<<S2.pop()<<endl;
}
```

QUESTION 2:
Implement the following stack operation using link list:

- **Push(x)**
- **Pop()**
- **Peek(position)**
- **Display()**

Create a stack structure and stack class for the above said implementation.

```
//STACK STRUCTURE
#include<iostream>
using namespace std;
int length=5;

struct node
{
   int data;
   node* next;
};
struct node*top;

void push(int data)
{
   node* t=new node;
   if(t==NULL)
      cout<<"stack overflow"<<endl;
   else
   {
      t->data=data;
      t->next=top;
      top=t;
   }
}
```

```
int pop()
{
        node* p;
        if(top==NULL)
        cout<<"stack is empty"<<endl;
        else
        p=top;
        top=top->next;
        p->next=NULL;
        delete p;

}

int peek(int pos)
{
    int i;
    node* p=top;
    for(i=0;i<length && p!=NULL;i++)
    {if(i==pos)
    cout<<p->data<<endl;
     p=p->next;
    }
}


void display()
{
    node* temp=top;
     if(top==NULL)
        cout<<"stack is empty"<<endl;
    else
    {
        while(temp!=NULL)
        {
            cout<<temp->data<<endl;
            temp=temp->next;
        }

    }

}
```

```
int main()
{
   int pos;
   push(10);
   push(20);
   push(30);
   push(40);
   push(50);
   push(60);
  push(70);

  display();
  pop();
  pop();
  cout<<"this is the stack after two pops"<<endl;
  display();
  cout<<"enter the position to know the data on it"<<endl;
  cin>>pos;
  peek(pos) ;
}
```

STACK CLASS

```
#include<iostream>
using namespace std;
int length=5;

class node
{public:
    int data;
    node* next;
};
class linklist
{

public:
 node* top;
 linklist()
 {
     top=NULL;
 }
void push(int data)
{
    node* t=new node;
```

```
    if(t==NULL)
        cout<<"stack overflow"<<endl;
    else
    {
        t->data=data;
        t->next=top;
        top=t;
    }
}
int pop()
{
    node* p=top;
     if(top==NULL)
        cout<<"stack is empty"<<endl;
        else
        top=top->next;
        p->next=NULL;
        delete p;
}

int peek(int pos)
{
    int i;
    node* p=top;
    for(i=0;i<length && p!=NULL;i++)
    {if(i==pos)
    cout<<p->data<<endl;
     p=p->next;
    }
}

void display()
{
    node* temp=top;
     if(top==NULL)
        cout<<"stack is empty"<<endl;
    else
    {
        while(temp!=NULL)
        {
            cout<<temp->data<<endl;
            temp=temp->next;
        }
```

```
    }
}
};
int main()
{
  int pos;
  linklist l1;
  l1.push(10);
  l1.push(20);
  l1.push(30);
  l1.push(40);
  l1.push(50);
  l1.push(60);
  l1.push(70);

  l1.display();
  l1.pop();
  l1.pop();
  cout<<"this is the stack after two pops"<<endl;
  l1.display();
  cout<<"enter the position to know the data on it"<<endl;
  cin>>pos;
  l1.peek(pos) ;
}
```

QUESTION 3:

Implement linear queue by writing a class and the following :

- **Insert queue**
- **Remove queue**
- **Isfull()**
- **isempty()**

Write main function to exemplify the results.Also write a main function to make the implementation a “Menu driven”.

```
#include<iostream>
using namespace std;
#define val 5 //macros

class Queue
{
  public:
  int arr[val];
  int frontend,rear;
  Queue();
  void insertq(int );
  int removeq();
  bool isfull();
  bool isempty();
};

Queue::Queue()
{
   frontend=rear=-1;
}

void Queue::insertq(int data)
{
  if (! isfull())
    {
  ++rear;
     }
   arr[rear]=data;
   cout<<"no inserted is:"<<arr[rear]<<endl;
}

int Queue::removeq()
   {
   ++frontend;
   int item=arr[frontend];
   cout<<item;
}

bool Queue::isfull()
{
return rear==val-1;
}
```

```
bool Queue::isempty()
{
  return rear==frontend;
}

int main()
{
int i,n,choice,y;
Queue obj;
do{
cout<<" 1.Insert queue"<<endl;
cout<<" 2.Remove queue"<<endl;
cout<<" 3.Value of rear when queue is full"<<endl;
cout<<" 4.Value of rear when queue is empty"<<endl;
cout<<" 5.Exit"<<endl;
cin>>choice;
switch(choice)
{
   case 1:
         for (i=1;i<val;i++)
         {
         cout<<"enter the nos"<<endl;
         cin>>n;
         obj.insertq(n);
         }
         break;

   case 2:
         obj.removeq();
         break
;
   case 3:
         cout<<obj.rear<<endl;
         break;

   case 4:
         cout<<obj.frontend<<endl;
         break;

   case 5:
      cout<<"Exiting"<<endl;
      break;
```

```
   default:
         cout<<"Invalid Choice"<<endl;
         break;
            }

      }
 while(choice!=5);
    return 0;
}
```

QUESTION 4:

- Implement priority queue by writing a class and the following :

- **Insert queue**
- **Remove queue**
- **Isfull()**
- **isempty()**

Write main function to exemplify the results.Also write a main function to make the implementation a "Menu driven".

```
#include <iostream>
using namespace std;
#define MAX 5 //macro
void insertq(int);
void deleteq(int);
void create();
void check(int);
void displayq();
int pqueue[MAX];
int frontend, rear;
```

```
int main()
{
   int n, ch;
   cout<<" Insert an element into queue"<<endl;
   cout<<" Delete an element from queue"<<endl;
   cout<<" Display queue elements"<<endl;
   cout<<" Exit"<<endl;

   create();
      while (1)
      {
      cout<<"Enter your choice : "<<endl;
      cin>>ch;

      switch (ch)
      {
      case 1:
         cout<<"Enter value to be inserted :"<<endl;
         cin>>n;
         insertq(n);
         break;

      case 2:
         cout<<"Enter value to delete : "<<endl;
         cin>>n;
         deleteq(n);
         break;

      case 3:
         displayq();
         break;

      case 4:
         cout<<"Exit"<<endl;
         break;
      default:
         cout<<"Choice is incorrect, Enter a correct choice"<<endl;
      }
   }
}
```

```
void create()
{
   frontend = rear = -1;
}


void insertq(int data)
{
   if (rear >= MAX - 1)
   {
      cout<<"Queue overflow no more elements can be inserted"<<endl;
      return;
   }
   if ((frontend == -1) && (rear == -1))
   {
      frontend++;
      rear++;
      pqueue[rear] = data;
      return;
   }
   else
      check(data);
   rear++;
}


void check(int data)
{
   int i,j;
   for (i= 0; i <= rear;pqueue [i++])
   {
      if (data >= pqueue[i])
      {
         for (j = rear + 1; j > i; j--)
         {
            pqueue[j] = pqueue[j - 1];
         }
         pqueue[i] = data;
         return;
      }
   }
   pqueue[i] = data;
}
```

```
void deleteq(int data)
{
   int i;

   if ((frontend==-1) && (rear==-1))
   {
      cout<<"Queue is empty no elements to delete"<<endl;
      return;
   }

   for (i = 0; i <= rear; i++)
   {
      if (data == pqueue[i])
      {
         for (; i < rear; i++)
         {
            pqueue[i] = pqueue[i + 1];
         }

      pqueue[i] = -99;
      rear--;
      if (rear == -1)
         frontend = -1;
          return;
      }
   }
}

void displayq()
{
      if ((frontend == -1) && (rear == -1))
      {
      cout<<"Queue is empty"<<endl;
      return;
      }
   for (; frontend <= rear; frontend++)
   {
      cout<<pqueue[frontend]<<endl;
   }

   frontend = 0;
}
```