Name: **Muhammad Anas Baig**

Enrollment No.: **01-134152-037**

Section: **BS(CS)-4A**

LAB-JOURNAL-3

Exercise 1:

Create a class Circular Queue that implements the functionality of a queue providing all the required operations (Enqueue(), Dequeue(), Empty(), Full() and getFront()).

Solution:

queue.h File:

```
#pragma once
class queue
{
public:
    int *arrqueue;
    int size, front, rear, count;
public:
    queue(int);
    bool isEmpty();
    bool isFull();
    void enqueue(int);
    int dequeue();
    int getFront();
    void display();
};
```

queue.cpp File:

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


queue::queue(int size)
{
    this->size = size;
    arrqueue = new int [this->size];
    rear = front = count = 0;
}

bool queue::isEmpty()
{
    if(count == 0)
    {
        return true;
    }
```

```cpp
    else
    {
        return false;
    }
}

bool queue::isFull()
{
    if(count == size)
    {
        return true;
    }
    else
    {
        return false;
    }
}

void queue::enqueue(int value)
{
    if(!isFull())
    {
        arrqueue[rear] = value;
        rear = (rear+1) % size;
        count++;
    }
    else
    {
        cout<<"Circular Queue Overflow"<<endl;
    }
}

int queue::dequeue()
{
    if(!isEmpty())
    {
        int temp = arrqueue[front];
        front = (front+1) % size;
        count--;
        return temp;
    }
    else
    {
        cout<<"Queue Underflow"<<endl;
        return (-1);
    }
}

int queue::getFront()
{
    if(!isEmpty())
    {
        return arrqueue[front];
    }
    else
    {
        cout<<"Queue Empty"<<endl;
        return (-1);
    }
}

```

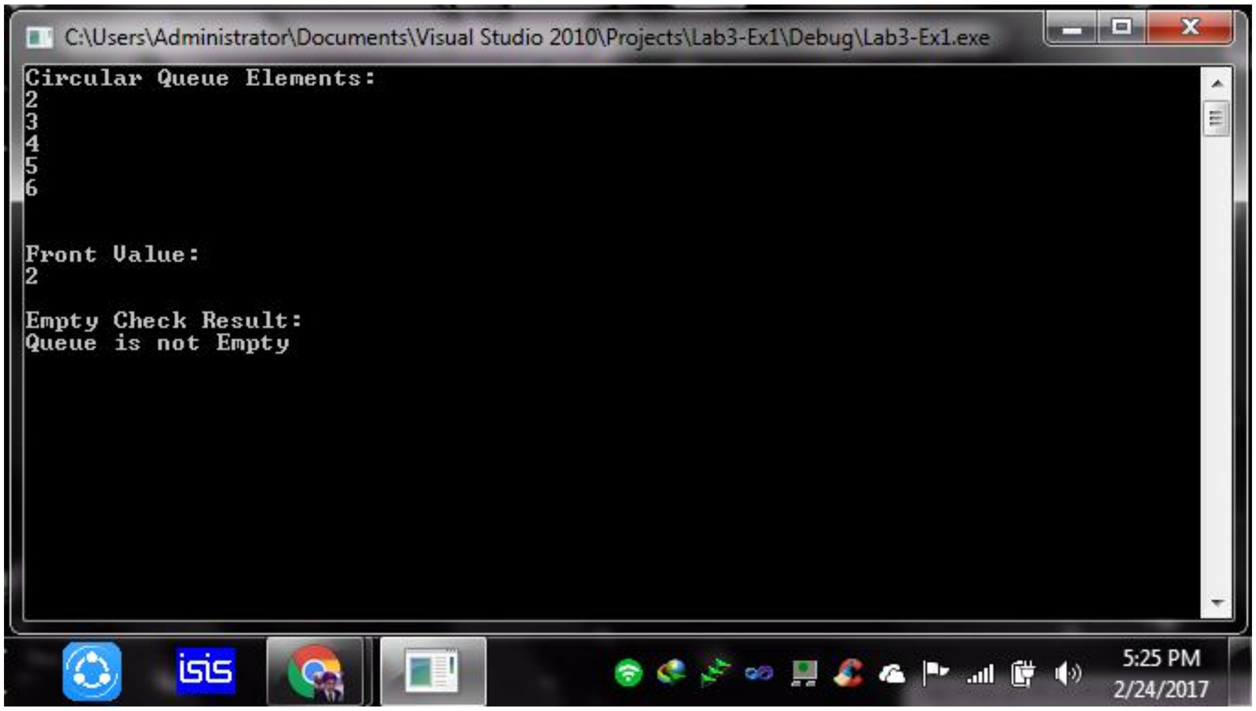
```
void queue::display()
{
    if(!isEmpty())
    {
        if(rear<=front)
        {
            for( int i=front; i<size; i++)
            {
                cout<<arrqueue[i]<<endl;
            }
            for( int i=0; i<=(rear-1); i++)
            {
                cout<<arrqueue[i]<<endl;
            }
        }
        else
        {
            for( int i=front; i<rear; i++)
            {
                cout<<arrqueue[i]<<endl;
                    }
                }
            }
            else
            {
                cout<<"Queue Empty"<<endl;
            }
        }
```

main.cpp File:

```
#include "queue.h"
#include "conio.h"
#include <iostream>
using namespace std;

void main()
{
    queue q(5);

    //after enqueing 5 elements, queue will become full at this stage
    q.enqueue(0);
    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);
    q.enqueue(4);
    //after enqueing 5 elements, queue will become full at this stage


    //first two elements of queue(0 & 1) are deleted
    q.dequeue();
    q.dequeue();
    //first two elements of queue(0 & 1) are deleted


    //two more elements are added to queue at this stage
    q.enqueue(5);
    q.enqueue(6);
    //two more elements are added to queue at this stage

```

```cpp

    //according to FIFO structure of queue, it should display elements which are entere
d first (2, 3, 4, 5, 6)
    cout<<"Circular Queue Elements:"<<endl;
    q.display();
    //according to FIFO structure of queue, it should display elements which are entere
d first (2, 3, 4, 5, 6)


    //getFront should display the value entered first in the queue
    cout<<endl;
    cout<<endl;
    cout<<"Front Value:"<<endl<<q.dequeue();
    //getFront should display the value entered first in the queue


    //Empty Check
    cout<<endl;
    cout<<endl;
    cout<<"Empty Check Result:"<<endl;
    if(q.isEmpty())
    {
        cout<<"Queue is Empty"<<endl;
    }
    else
    {
        cout<<"Queue is not Empty"<<endl;
    }
    //Empty Check

    getch();
}
```

Output:


C:\Users\Administrator\Documents\Visual Studio 2010\Projects\Lab3-Ex1\Debug\Lab3-Ex1.exe

```
Circular Queue Elements:
2
3
4
5
6


Front Value:
2

Empty Check Result:
Queue is not Empty
```

Exercise 2:

Write a program that reads a string from a text file and determines if the string is a palindrome or not. Use a Stack and a Queue to check for the palindrome.

Solution:

stack.h File:

```cpp
#pragma once
class stack
{
private:
    char *arrStack;
    int top;
    int size;
public:
    stack(void);
    stack(int);
    bool isEmpty();
    bool isFull();
    void push(char);
    char pop();
    char _top();
    void display();
};
```

stack.cpp File:

```cpp
#include "stack.h"
#include <iostream>
using namespace std;

stack::stack(void)
{
    size = 100;
    arrStack = new char [size];
    top = -1;
}

stack::stack(int size)
{
    this->size = size;
    arrStack = new char [this->size];
    top = -1;
}

bool stack::isEmpty()
{
    if(top == -1)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool stack::isFull()
{
    if(top == (size-1))
    {
        return true;
    }
    else
    {
        return false;
    }
}

void stack::push(char value)
{
    if(!isFull())
    {
        arrStack[++top] = value;
    }
    else
    {
        cout<<"Stack Overflow!!!"<<endl;
    }
}

char stack::pop()
{
    if(!isEmpty())
    {
        return (arrStack[top--]);
    }
    else
    {
        cout<<"Stack Underflow!!!"<<endl;
    }
}

char stack::_top()
{
    if(!isEmpty())
    {
        return arrStack[top];
    }
    else
    {
        cout<<"Stack Empty!!!";
    }
}

void stack::display()
{
    for(int i=top; i>=0; i--)
    {
        cout<<arrStack[i];
    }
}
```

queue.h File:

```cpp
#pragma once
class queue
{
private:
    char *arrqueue;
    char front, rear, size;
public:
    queue(char);
    bool isEmpty();
    bool isFull();
    void enqueue(char);
    char dequeue();
    char getFront();
    void display();
};
```

queue.cpp File:

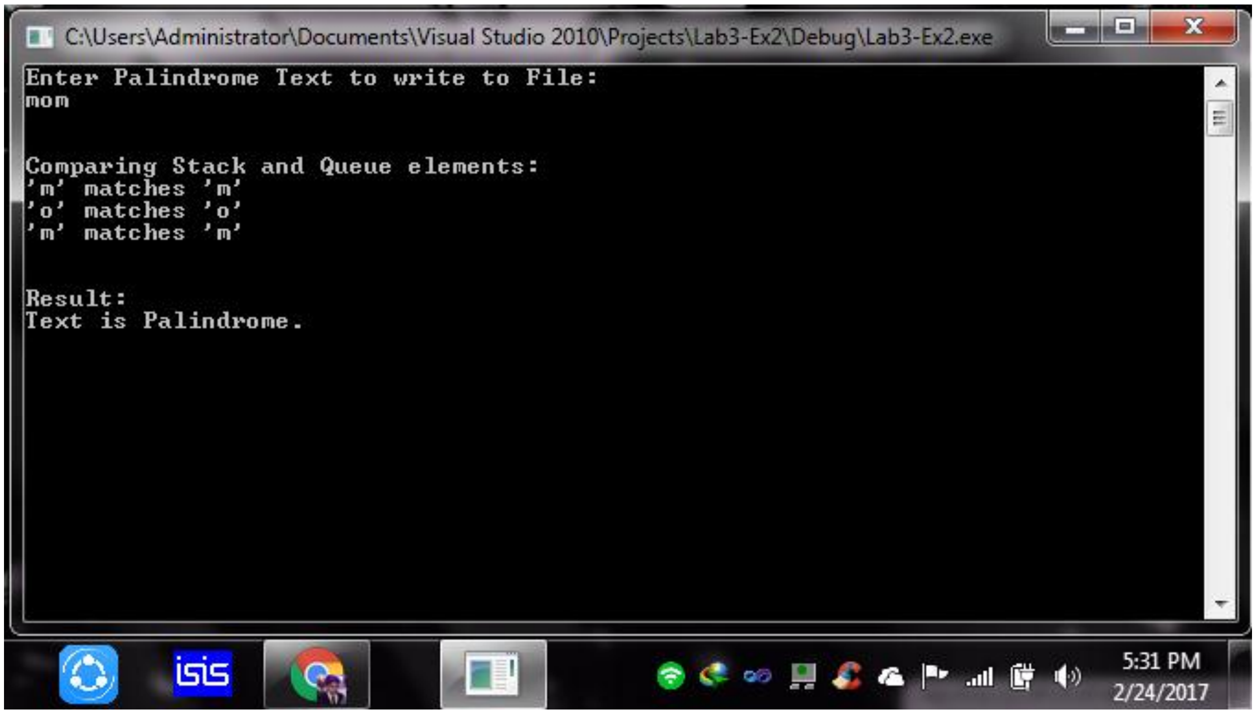
```cpp
#include "queue.h"
#include <iostream>
using namespace std;


queue::queue(char size)
{
    this->size = size;
    arrqueue = new char [this->size];
    rear = front = -1;
}

bool queue::isEmpty()
{
    if(front == rear)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool queue::isFull()
{
    if(rear == (size-1))
    {
        return true;
    }
    else
    {
        return false;
    }
}

void queue::enqueue(char value)
{
    if(!isFull())
    {
        arrqueue [++rear] = value;
    }
    else
    {
        cout<<"Queue Overflow!!!"<<endl;
    }
}

char queue::dequeue()
{
    if(!isEmpty())
    {
        return arrqueue [++front];
    }
    else
    {
        cout<<"Queue Empty!!!"<<endl;
        return (-1);
    }
}

char queue::getFront()
{
    if(!isEmpty())
    {
        return arrqueue [front +1];
    }
    else
    {
        cout<<"Queue Empty!!!"<<endl;
        return (-1);
    }
}

void queue::display()
{
    if(!isEmpty())
    {
        for(char i=(front + 1); i<=rear; i++)
        {
            cout<<arrqueue[i]<<endl;
        }
    }
    else
    {
        cout<<"Queue Empty!!!"<<endl;
    }
}
```

main.cpp File:

```cpp
#include "queue.h"
#include "stack.h"
#include "conio.h"
#include "string.h"
#include "string"
#include <fstream>
#include <iostream>
using namespace std;
```

```cpp

void main()
{
    //=========Module to get Palindrome Text from user and then saving to file===========
    string line_write; //string to get input and then write text to file
    cout<<"Enter Palindrome Text to write to File: "<<endl;
    getline(cin, line_write); //getting input from user
    ofstream write_file ("palindrome_text.txt"); //write file object
    write_file<<line_write; //writing string whole text to file
    write_file.close(); //closing write file

    //=========Module to read Palindrome Text from file==================================
    string line_read; //string to read text from file
    ifstream file_read ("palindrome_text.txt"); //read file object
    getline(file_read,line_read); //reading whole text from file to string
    file_read.close(); //closing read file

    //=========Module to compare Stack and Queue elements================================
    int check=1; //it will be 1 for palindrome and  0 from non-palindrome
    stack s(line_read.length()); //stack to save string text
    queue q(line_read.length()); //stack to save string text
    for(int i=0; i<line_read.length(); i++) //loop that stores each character of string to stack and queue
    {
        s.push(line_read[i]); //pushing to stack
        q.enqueue(line_read[i]); //enqueing to queue
    }
    cout<<endl;
    cout<<endl;
    cout<<"Comparing Stack and Queue elements:"<<endl;
    while(!s.isEmpty() && !q.isEmpty()) //loop that compares stack and queue elements one by one
    {
        char item_1 = q.dequeue(); //to get queue element
        char item_2 = s.pop(); //to get stack element
        if(item_1 == item_2) //if both characters matches
        {
            cout<<"'"<<item_1<<"'"<<" matches "<<"'"<<item_2<<"'"<<endl;
        }
        else
        {
            cout<<"'"<<item_1<<"'"<<" doesn't matches "<<"'"<<item_2<<"'"<<endl;
            check = 0;
        }
    }
    cout<<endl;
    cout<<endl;
    cout<<"Result:"<<endl;
    if(check == 0) //check is 0 for non-palindrome and 1 for palindrome
    {
        cout<<"Text is not Palindrome."<<endl;
    }
    else
    {
        cout<<"Text is Palindrome."<<endl;
    }

    getch();
}
```

Output:


C:\Users\Administrator\Documents\Visual Studio 2010\Projects\Lab3-Ex2\Debug\Lab3-Ex2.exe
5:31 PM
2/24/2017

```
Enter Palindrome Text to write to File:
mom

Comparing Stack and Queue elements:
'm' matches 'm'
'o' matches 'o'
'm' matches 'm'


Result:
Text is Palindrome.
```

Exercise 3:

Implement an ascending priority queue using a (linear) array. The items can be added to the queue like a normal queue. However, the dequeue operation should remove the minimum item from the queue. Once an element is removed from the queue, shift the rest of the elements to the left. A sample run of the program should result in the contents as illustrated in the following Figure. (You do not need the maker 'front' for this implementation).

Solution:

queue.h File:

```
#pragma once
class queue
{
private:
    int *arrqueue;
    int front, rear, size;
public:
    queue(int);
    bool isEmpty();
    bool isFull();
    void enqueue(int);
    int dequeue();
    int getFront();
    void display();
};
```

queue.cpp File:

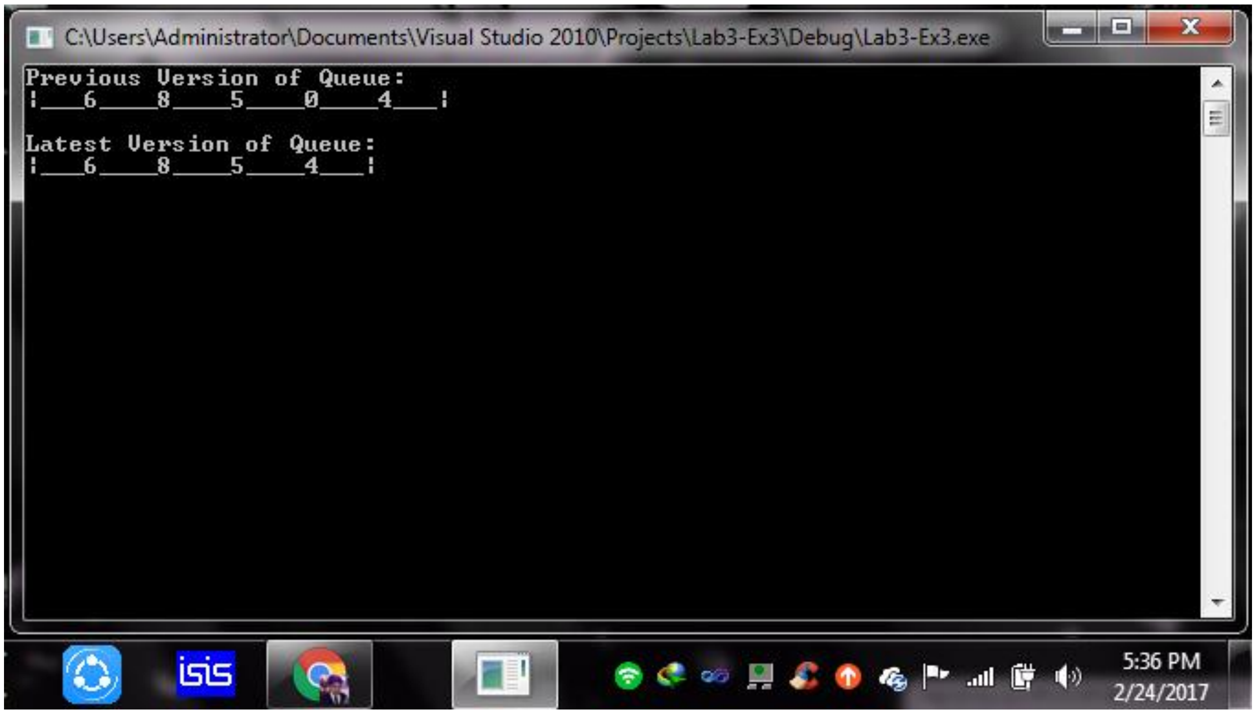
```
#include "queue.h"
#include <iostream>
using namespace std;


queue::queue(int size)
{
    this->size = size;
    arrqueue = new int [this->size];
    rear = front = -1;
}

bool queue::isEmpty()
{
    if(front == rear)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool queue::isFull()
{
    if(rear == (size-1))
    {
        return true;
    }
    else
    {
        return false;
    }
}

void queue::enqueue(int value)
{
    if(!isFull())
    {
        arrqueue [++rear] = value;
    }
    else
    {
        cout<<"Queue Overflow!!!"<<endl;
    }
}

int queue::dequeue()
{
    if(!isEmpty())
    {
        int min = arrqueue[0]; //declaring 1st element to minimum

        for(int i=1; i<rear; i++)
        {
            if(arrqueue[i] < min) //logic to find minimum
            {
                min = arrqueue[i];
            }
        }

        for(int j=0; j<rear; j++)
        {
            if(arrqueue[j] == min)
            {
                for(int k=j; j<rear; j++) //shifting elements to left
                {
                    arrqueue[j] = arrqueue[j+1];
                }
            }
        }
        rear--; //decrementing rear
    }
    else
    {
        cout<<"Queue Empty!!!"<<endl;
        return (-1);
    }
}

int queue::getFront()
{
    if(!isEmpty())
    {
        return arrqueue [front +1];
    }
    else
    {
        cout<<"Queue Empty!!!"<<endl;
        return (-1);
    }
}

void queue::display()
{
    if(!isEmpty())
    {
        cout<<"|_";
            for(int i=(front + 1); i<=rear; i++)
            {
                cout<<"__"<<arrqueue[i]<<"__";
            }
            cout<<"_|";
        }
        else
        {
            cout<<"Queue Empty!!!"<<endl;
        }
    }
```

main.cpp File:

```
#include "queue.h"
```

```cpp
#include "conio.h"
#include <iostream>
using namespace std;

void  main()
{
	queue q(10); //queue of 10 elements

	q.enqueue(6);
	q.enqueue(8);
	q.enqueue(5);
	q.enqueue(0);
	q.enqueue(4);

	cout<<"Previous Version of Queue:"<<endl;
	q.display();

	q.dequeue(); //dequeuing element

	cout<<endl;
	cout<<endl;
	cout<<"Latest Version of Queue:"<<endl;
	q.display();

	getch();
}
```

Output:



C:\Users\Administrator\Documents\Visual Studio 2010\Projects\Lab3-Ex3\Debug\Lab3-Ex3.exe

```
Previous Version of Queue:
|___6_____8_____5_____0_____4___|

Latest Version of Queue:
|___6_____8_____5_____4___|
```