

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:

```
1. #pragma once
2. class queue
3. {
4. public:
5.     int *arrqueue;
6.     int size, front, rear, count;
7. public:
8.     queue(int);
9.     bool isEmpty();
10.    bool isFull();
11.    void enqueue(int);
12.    int dequeue();
13.    int getFront();
14.    void display();
15. };
```

#### queue.cpp File:

```
1. #include "queue.h"
2. #include <iostream>
3. using namespace std;
4.
5.
6. queue::queue(int size)
7. {
8.     this->size = size;
9.     arrqueue = new int [this->size];
10.    rear = front = count = 0;
11. }
12.
13. bool queue::isEmpty()
14. {
15.     if(count == 0)
16.     {
17.         return true;
18.     }
```

```
19.     else
20.     {
21.         return false;
22.     }
23. }
24.
25. bool queue::isFull()
26. {
27.     if(count == size)
28.     {
29.         return true;
30.     }
31.     else
32.     {
33.         return false;
34.     }
35. }
36.
37. void queue::enqueue(int value)
38. {
39.     if(!isFull())
40.     {
41.         arrqueue[rear] = value;
42.         rear = (rear+1) % size;
43.         count++;
44.     }
45.     else
46.     {
47.         cout<<"Circular Queue Overflow"<<endl;
48.     }
49. }
50.
51. int queue::dequeue()
52. {
53.     if(!isEmpty())
54.     {
55.         int temp = arrqueue[front];
56.         front = (front+1) % size;
57.         count--;
58.         return temp;
59.     }
60.     else
61.     {
62.         cout<<"Queue Underflow"<<endl;
63.         return (-1);
64.     }
65. }
66.
67. int queue::getFront()
68. {
69.     if(!isEmpty())
70.     {
71.         return arrqueue[front];
72.     }
73.     else
74.     {
75.         cout<<"Queue Empty"<<endl;
76.         return (-1);
77.     }
78. }
79.
```

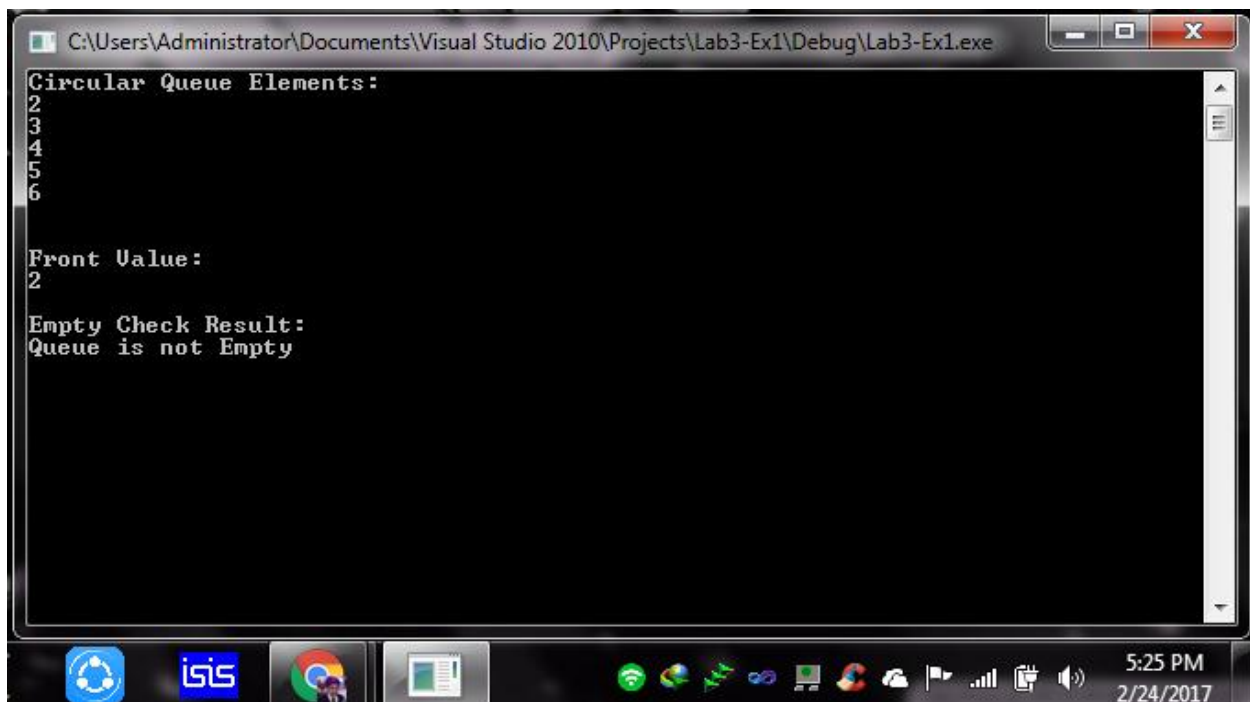
```
80. void queue::display()
81. {
82.     if(!isEmpty())
83.     {
84.         if(rear<=front)
85.         {
86.             for( int i=front; i<size; i++)
87.             {
88.                 cout<<arrqueue[i]<<endl;
89.             }
90.             for( int i=0; i<=(rear-1); i++)
91.             {
92.                 cout<<arrqueue[i]<<endl;
93.             }
94.         }
95.         else
96.         {
97.             for( int i=front; i<rear; i++)
98.             {
99.                 cout<<arrqueue[i]<<endl;
100.            }
101.        }
102.    }
103.    else
104.    {
105.        cout<<"Queue Empty"<<endl;
106.    }
107. }
```

### main.cpp File:

```
1. #include "queue.h"
2. #include "conio.h"
3. #include <iostream>
4. using namespace std;
5.
6. void main()
7. {
8.     queue q(5);
9.
10.    //after enqueueing 5 elements, queue will become full at this stage
11.    q.enqueue(0);
12.    q.enqueue(1);
13.    q.enqueue(2);
14.    q.enqueue(3);
15.    q.enqueue(4);
16.    //after enqueueing 5 elements, queue will become full at this stage
17.
18.
19.    //first two elements of queue(0 & 1) are deleted
20.    q.dequeue();
21.    q.dequeue();
22.    //first two elements of queue(0 & 1) are deleted
23.
24.
25.    //two more elements are added to queue at this stage
26.    q.enqueue(5);
27.    q.enqueue(6);
28.    //two more elements are added to queue at this stage
29. }
```

```
30.
31. //according to FIFO structure of queue, it should display elements which are entered first (2, 3, 4, 5, 6)
32. cout<<"Circular Queue Elements:"<<endl;
33. q.display();
34. //according to FIFO structure of queue, it should display elements which are entered first (2, 3, 4, 5, 6)
35.
36.
37. //getFront should display the value entered first in the queue
38. cout<<endl;
39. cout<<endl;
40. cout<<"Front Value:"<<endl<<q.dequeue();
41. //getFront should display the value entered first in the queue
42.
43.
44. //Empty Check
45. cout<<endl;
46. cout<<endl;
47. cout<<"Empty Check Result:"<<endl;
48. if(q.isEmpty())
49. {
50.     cout<<"Queue is Empty"<<endl;
51. }
52. else
53. {
54.     cout<<"Queue is not Empty"<<endl;
55. }
56. //Empty Check
57.
58. getch();
59. }
```

## 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:

```
1. #pragma once
2. class stack
3. {
4. private:
5.     char *arrStack;
6.     int top;
7.     int size;
8. public:
9.     stack(void);
10.    stack(int);
11.    bool isEmpty();
12.    bool isFull();
13.    void push(char);
14.    char pop();
15.    char _top();
16.    void display();
17. };
```

### stack.cpp File:

```
1. #include "stack.h"
2. #include <iostream>
3. using namespace std;
4.
5. stack::stack(void)
6. {
7.     size = 100;
8.     arrStack = new char [size];
9.     top = -1;
10. }
11.
12. stack::stack(int size)
13. {
14.     this->size = size;
15.     arrStack = new char [this->size];
16.     top = -1;
17. }
18.
19. bool stack::isEmpty()
20. {
21.     if(top == -1)
22.     {
23.         return true;
24.     }
25.     else
26.     {
27.         return false;
28.     }
```

```
29. }
30.
31. bool stack::isFull()
32. {
33.     if(top == (size-1))
34.     {
35.         return true;
36.     }
37.     else
38.     {
39.         return false;
40.     }
41. }
42.
43. void stack::push(char value)
44. {
45.     if(!isFull())
46.     {
47.         arrStack[++top] = value;
48.     }
49.     else
50.     {
51.         cout<<"Stack Overflow!!!"<<endl;
52.     }
53. }
54.
55. char stack::pop()
56. {
57.     if(!isEmpty())
58.     {
59.         return (arrStack[top--]);
60.     }
61.     else
62.     {
63.         cout<<"Stack Underflow!!!"<<endl;
64.     }
65. }
66.
67. char stack::_top()
68. {
69.     if(!isEmpty())
70.     {
71.         return arrStack[top];
72.     }
73.     else
74.     {
75.         cout<<"Stack Empty!!!";
76.     }
77. }
78.
79. void stack::display()
80. {
81.     for(int i=top; i>=0; i--)
82.     {
83.         cout<<arrStack[i];
84.     }
85. }
```

### queue.h File:

```
1. #pragma once
2. class queue
3. {
4. private:
5.     char *arrqueue;
6.     char front, rear, size;
7. public:
8.     queue(char);
9.     bool isEmpty();
10.    bool isFull();
11.    void enqueue(char);
12.    char dequeue();
13.    char getFront();
14.    void display();
15. };
```

### queue.cpp File:

```
1. #include "queue.h"
2. #include <iostream>
3. using namespace std;
4.
5.
6. queue::queue(char size)
7. {
8.     this->size = size;
9.     arrqueue = new char [this->size];
10.    rear = front = -1;
11. }
12.
13. bool queue::isEmpty()
14. {
15.     if(front == rear)
16.     {
17.         return true;
18.     }
19.     else
20.     {
21.         return false;
22.     }
23. }
24.
25. bool queue::isFull()
26. {
27.     if(rear == (size-1))
28.     {
29.         return true;
30.     }
31.     else
32.     {
33.         return false;
34.     }
35. }
36.
37. void queue::enqueue(char value)
38. {
39.     if(!isFull())
```

```
40.     {
41.         arrqueue[++rear] = value;
42.     }
43.     else
44.     {
45.         cout<<"Queue Overflow!!!"<<endl;
46.     }
47. }
48.
49. char queue::dequeue()
50. {
51.     if(!isEmpty())
52.     {
53.         return arrqueue[++front];
54.     }
55.     else
56.     {
57.         cout<<"Queue Empty!!!"<<endl;
58.         return (-1);
59.     }
60. }
61.
62. char queue::getFront()
63. {
64.     if(!isEmpty())
65.     {
66.         return arrqueue[front + 1];
67.     }
68.     else
69.     {
70.         cout<<"Queue Empty!!!"<<endl;
71.         return (-1);
72.     }
73. }
74.
75. void queue::display()
76. {
77.     if(!isEmpty())
78.     {
79.         for(char i=(front + 1); i<=rear; i++)
80.         {
81.             cout<<arrqueue[i]<<endl;
82.         }
83.     }
84.     else
85.     {
86.         cout<<"Queue Empty!!!"<<endl;
87.     }
88. }
```

### **main.cpp File:**

```
1. #include "queue.h"
2. #include "stack.h"
3. #include "conio.h"
4. #include "string.h"
5. #include "string"
6. #include <fstream>
7. #include <iostream>
8. using namespace std;
```



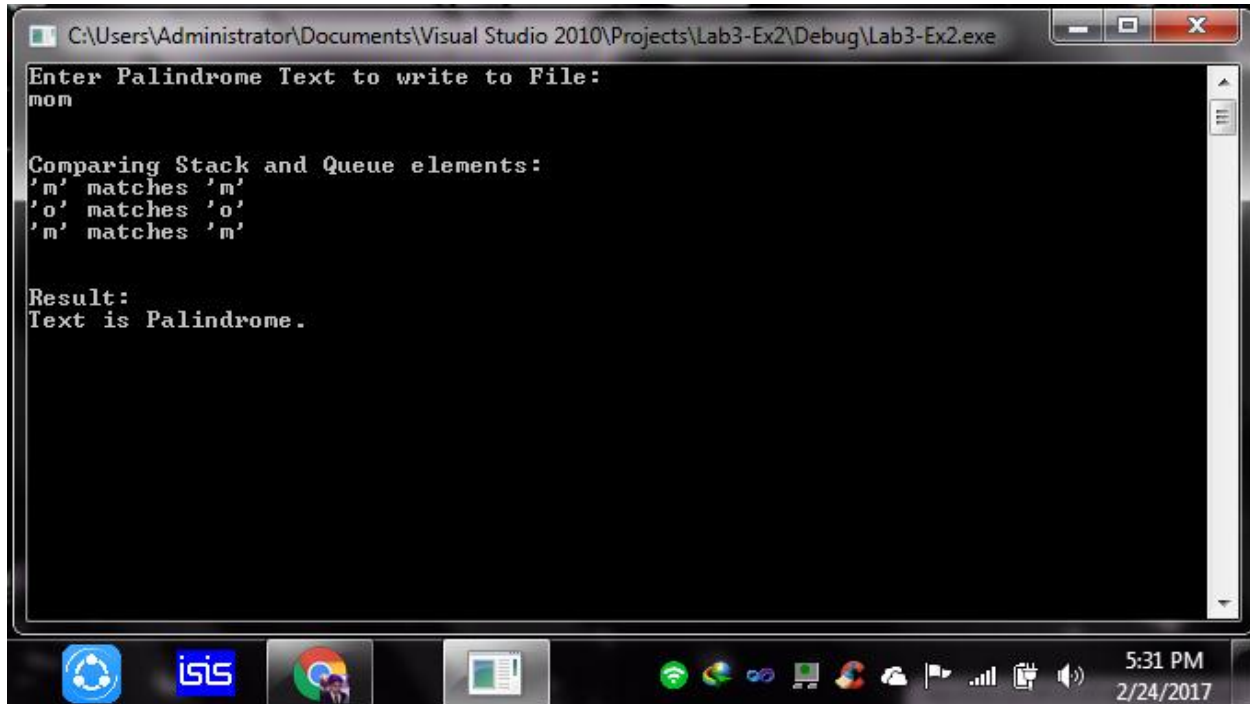
```

9.
10. void main()
11. {
12.     //=====Module to get Palindrome Text from user and then saving to file=====
    ==
13.     string line_write; //string to get input and then write text to file
14.     cout<<"Enter Palindrome Text to write to File: "<<endl;
15.     getline(cin, line_write); //getting input from user
16.     ofstream write_file ("palindrome_text.txt"); //write file object
17.     write_file<<line_write; //writing string whole text to file
18.     write_file.close(); //closing write file
19.
20.     //=====Module to read Palindrome Text from file=====
    ==
21.     string line_read; //string to read text from file
22.     ifstream file_read ("palindrome_text.txt"); //read file object
23.     getline(file_read,line_read); //reading whole text from file to string
24.     file_read.close(); //closing read file
25.
26.     //=====Module to compare Stack and Queue elements=====
    ==
27.     int check=1; //it will be 1 for palindrome and 0 from non-palindrome
28.     stack s(line_read.length()); //stack to save string text
29.     queue q(line_read.length()); //stack to save string text
30.     for(int i=0; i<line_read.length(); i++) //loop that stores each character of string
        to stack and queue
31.     {
32.         s.push(line_read[i]); //pushing to stack
33.         q.enqueue(line_read[i]); //enqueueing to queue
34.     }
35.     cout<<endl;
36.     cout<<endl;
37.     cout<<"Comparing Stack and Queue elements:"<<endl;
38.     while(!s.isEmpty() && !q.isEmpty()) //loop that compares stack and queue elements o
        ne by one
39.     {
40.         char item_1 = q.dequeue(); //to get queue element
41.         char item_2 = s.pop(); //to get stack element
42.         if(item_1 == item_2) //if both characters matches
43.         {
44.             cout<<" "<<item_1<<" "<<" matches "<<" "<<item_2<<" "<<endl;
45.         }
46.         else
47.         {
48.             cout<<" "<<item_1<<" "<<" doesn't matches "<<" "<<item_2<<" "<<endl;
49.             check = 0;
50.         }
51.     }
52.     cout<<endl;
53.     cout<<endl;
54.     cout<<"Result:"<<endl;
55.     if(check == 0) //check is 0 for non-palindrome and 1 for palindrome
56.     {
57.         cout<<"Text is not Palindrome."<<endl;
58.     }
59.     else
60.     {
61.         cout<<"Text is Palindrome."<<endl;
62.     }
63.
64.     getch();

```

```
65. }
```

## Output:



```
C:\Users\Administrator\Documents\Visual Studio 2010\Projects\Lab3-Ex2\Debug\Lab3-Ex2.exe
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:

```
1. #pragma once
2. class queue
3. {
4. private:
5.     int *arrqueue;
6.     int front, rear, size;
7. public:
8.     queue(int);
9.     bool isEmpty();
10.    bool isFull();
11.    void enqueue(int);
12.    int dequeue();
```

```
13.     int getFront();
14.     void display();
15. };
```

### **queue.cpp File:**

```
1. #include "queue.h"
2. #include <iostream>
3. using namespace std;
4.
5.
6. queue::queue(int size)
7. {
8.     this->size = size;
9.     arrqueue = new int [this->size];
10.    rear = front = -1;
11. }
12.
13. bool queue::isEmpty()
14. {
15.     if(front == rear)
16.     {
17.         return true;
18.     }
19.     else
20.     {
21.         return false;
22.     }
23. }
24.
25. bool queue::isFull()
26. {
27.     if(rear == (size-1))
28.     {
29.         return true;
30.     }
31.     else
32.     {
33.         return false;
34.     }
35. }
36.
37. void queue::enqueue(int value)
38. {
39.     if(!isFull())
40.     {
41.         arrqueue [++rear] = value;
42.     }
43.     else
44.     {
45.         cout<<"Queue Overflow!!!"<<endl;
46.     }
47. }
48.
49. int queue::dequeue()
50. {
51.     if(!isEmpty())
52.     {
53.         int min = arrqueue[0]; //declaring 1st element to minimum
54.     }
```

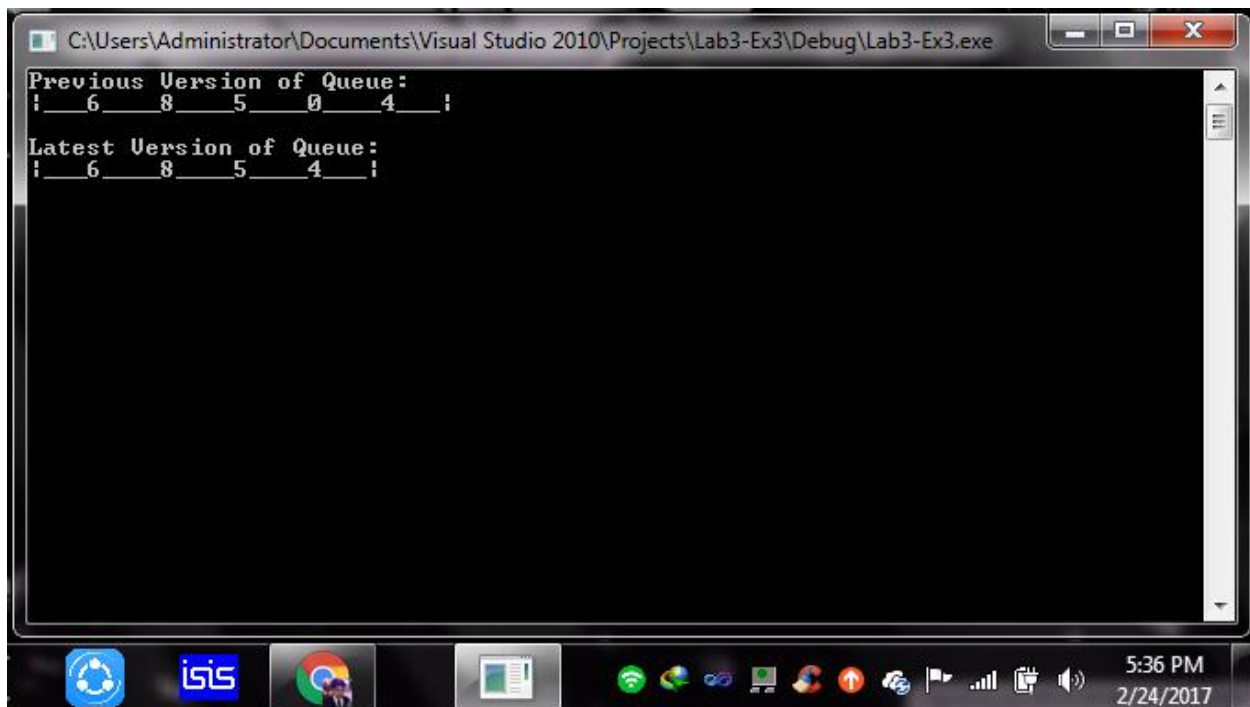
```
55.     for(int i=1; i<rear; i++)
56.     {
57.         if(arrqueue[i] < min) //logic to find minimum
58.         {
59.             min = arrqueue[i];
60.         }
61.     }
62.
63.     for(int j=0; j<rear; j++)
64.     {
65.         if(arrqueue[j] == min)
66.         {
67.             for(int k=j; k<rear; k++) //shifting elements to left
68.             {
69.                 arrqueue[j] = arrqueue[j+1];
70.             }
71.         }
72.     }
73.     rear--; //decrementing rear
74. }
75. else
76. {
77.     cout<<"Queue Empty!!!"<<endl;
78.     return (-1);
79. }
80. }
81.
82. int queue::getFront()
83. {
84.     if(!isEmpty())
85.     {
86.         return arrqueue [front +1];
87.     }
88.     else
89.     {
90.         cout<<"Queue Empty!!!"<<endl;
91.         return (-1);
92.     }
93. }
94.
95. void queue::display()
96. {
97.     if(!isEmpty())
98.     {
99.         cout<<"|_";
100.        for(int i=(front + 1); i<=rear; i++)
101.        {
102.            cout<<"_"<<arrqueue[i]<<"_";
103.        }
104.        cout<<"_|";
105.    }
106.    else
107.    {
108.        cout<<"Queue Empty!!!"<<endl;
109.    }
110. }
```

### main.cpp File:

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

```
2. #include "conio.h"
3. #include <iostream>
4. using namespace std;
5.
6. void main()
7. {
8.     queue q(10); //queue of 10 elements
9.
10.    q.enqueue(6);
11.    q.enqueue(8);
12.    q.enqueue(5);
13.    q.enqueue(0);
14.    q.enqueue(4);
15.
16.    cout<<"Previous Version of Queue:"<<endl;
17.    q.display();
18.
19.    q.dequeue(); //dequeuing element
20.
21.    cout<<endl;
22.    cout<<endl;
23.    cout<<"Latest Version of Queue:"<<endl;
24.    q.display();
25.
26.    getch();
27. }
```

## 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 !
```