Experiment #3

Static Linear Queue

Student's Name:	
Semester:	Date:

Assessment:

Assessment Point	**Weight**	**Grade**
Methodology and correctness of results		
Discussion of results		
Participation		
Assessment Points' Grade:		

Comments:

Experiment #3:

Static Linear Queue in C++ Programming Language

Objectives:

1. To introduce the students with the concept of linear queues
2. To implement static linear queues
3. To implement different functions of linear queues
4. To understand the disadvantages of array implementation of linear queues

Discussion:

Linear queues are abstract data types with FIFO (First In First Out) type of working, where a new element is added at the back of the linear queue (rear) and an element is removed from that front of the linear queue (front).

The functions associated with linear queue are:
1. isempty() – Returns whether the linear queue is empty – Time Complexity : O(1)
2. isfull – Returns whether the linear queue is full – Time Complexity : O(1)
3. Enqueue (g) – Adds the element 'g' at the end of the queue – Time Complexity:O(1)
4. Dequeue() – Return the elements – Time Complexity : O(1)

Static Linear Queue implementation

```
// this program for implementing a static linear queue using templates
// programmed by Dr.Aryaf Al-adwan
#include<iostream>
Using namespace std;
const int size=8;
template <class T>
class linearqueue
{
private:
        int front;
        int rear;
        int count;
        T arr[size];
```

```
public:
        linearqueue()
        {
                front=-1;
                rear=-1;
                count=0;
        }

        bool isEmpty()
        {
                if((rear==-1 && front==-1)|| front==size-1)
                        return true;
                else
                        return false;
        }

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

        void enqueue (T item)
        {
                if(isFull())
                {
                 cout<<"Queue is Full\n";
                }
                else
                {
                        rear++;
                        arr[rear]=item;
                        count++;
                }
        }

        T dequeue ()
        {
                T dequeueitem;
                if(isEmpty())
                {
                  cout<<"Queue is Empty\n";
                        return 0;
                }
                else
                {
                        front++;
                        dequeueitem=arr[front];
                        count--;
                        if(count<0)
                                count=0;
                        return dequeueitem;
                }
        }
};

void main()
{
        linearqueue <int> q1;
        q1.dequeue();
        q1.enqueue(1);
       q1.enqueue(2);
        q1.enqueue(3);
        q1.enqueue(4);
        q1.enqueue(5);
        q1.enqueue(6);
        cout<<"number of items in the queue: "<<q1.numberofitems()<<endl;
        for(int i=0;i<size;i++)
            cout << q1.dequeue() << endl;
        q1.print_rearandfront();
        cout<<"number of items in the queue: "<<q1.numberofitems()<<endl;
        q1.enqueue(77);
        q1.print_rearandfront();
        q1.enqueue(88);
        q1.print_rearandfront();
        q1.enqueue(99);
        cout<<"number of items in the queue: "<<q1.numberofitems()<<endl;
        q1.print_queue();
        q1.print_rearandfront();
}
```

Exercise 1:

Write a c++ program to print the contents of the queue?

Solution to Exercise 1

Output

Exercise 2:

Write a c++ program to print the number of elements in the queue?

Solution to Exercise 2

Output

Exercise 3:

Write a c++ program to search for an element stored in the queue?

Solution to Exercise 3

Output

Exercise 4:

Write a C++ program to count the number of occurrences of given number stored in a static linear queue.

Solution to Exercise 3

Output