Experiment #4

Static Circular 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 #4:

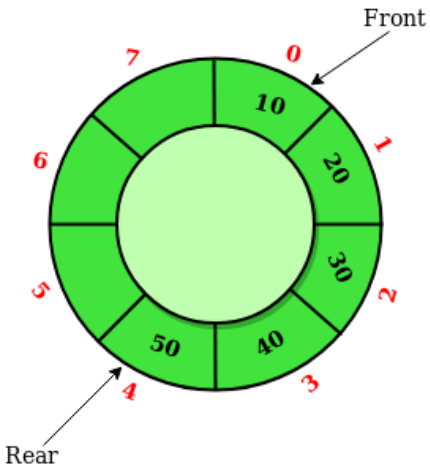
Static Circular Queue in C++ Programming Language

Objectives:

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

Discussion:

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **'Ring Buffer'**.



In a linear Queue, we can insert elements until queue becomes full. But once queue becomes full, we cannot insert the next element even if there is a space in front of queue. So, overwrite can be done in circular queue.

Static Circular Queue implementation

```
// Static Circular Queue using count variable to determine full and empty queue
// programmed by Dr.Aryaf Al-adwan

#include < iostream >
using namespace std;
const int size =5;
template<class T>
```

```
class CircularQ
{
private:
T q[size];
int front,rear;
int count;
public:

CircularQ()
{
front = 0;
rear = size - 1;
count = 0;
}

bool isFull()
{
        return(count==size);
}

bool isEmpty()
{
        return (count==0);
}

void enqueue(T item)
{
if(isFull())
{
        cout<<"\nfull \n";
}
```

```
else
{
rear = (rear + 1) % size;
                q[rear] = item;
                count++;
}
}

int dequeue()
{
        if(isEmpty())
        {
                cout<<"empty";
                return 0;
        }
        else
        {
int x = q[front];
front = (front + 1) % size;
count--;
return x;
        }
}
};

int main()
{
CircularQ <int> qq;
cout<<qq.dequeue();
qq.enqueue(6);
qq.enqueue(2);
qq.enqueue(8);
qq.enqueue(1);
```

```
qq.enqueue(5);
qq.enqueue(7);
while (qq.getSize())
cout << qq.dequeue() << endl;
qq.enqueue(10);
qq.print_queue();
return 0;
}
```

Exercise 1:

Write a c++ program to print the contents of the circular 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 find the sum of all elements stored in the circular queue?

Solution to Exercise 3

.

Output