## Circular Queue

In a normal Queue Data Structure, we can insert elements until queue becomes full. But once the queue becomes full, we cannot insert the next element until all the elements are deleted from the queue. For example, consider the queue below:

The queue after inserting all the elements into it is as follows:

**21**

**11**

**52**

**54**

**79**

**74**

**front**

**rear**

Now consider the following situation after deleting three elements from the queue:

**54**

**79**

**74**

**front**

**rear**

**Queue is full (even if three elements are deleted from front)**

This situation also says that Queue is Full and we cannot insert the new element because **'rear'** is still at last position. In the above situation, even though we have empty positions in the queue we cannot make use of them to insert the new element. This is the major problem in a normal queue data structure. To overcome this problem, we use a **circular queue** data structure.

A Circular Queue can be defined as follows:

A 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.

Graphical representation of a circular queue is as follows:

###### Front

###### Rear