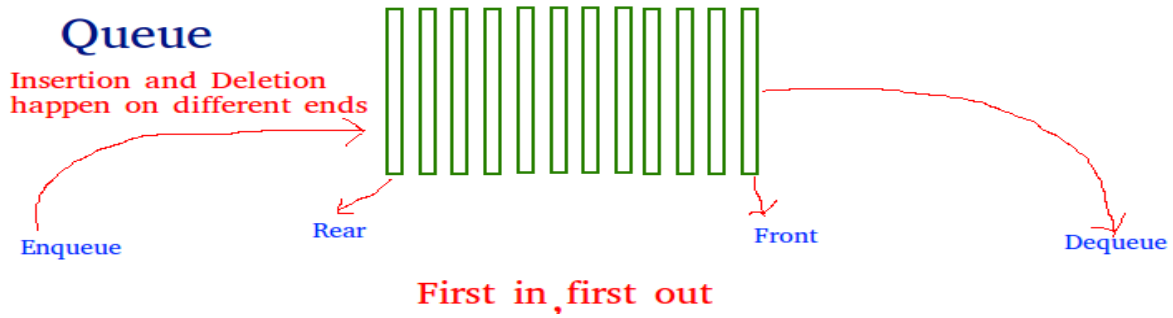
Queue and Deque

Queue

Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.



The following two main operations must be implemented efficiently. In a Queue data structure, we maintain two pointers, *front* and *rear*. The *front* points the first item of queue and *rear* points to last item.

enQueue() This operation adds a new node after *rear* and moves *rear* to the next node.

deQueue() This operation removes the front node and moves *front* to the next node.

```
using System;
 class QNode {
```

```
    public int key;
    public QNode next;
    public QNode(int key)
    {
        this.key = key;
        this.next = null;
    }
}
class Queue {
    QNode front, rear;


    public Queue()
    {
        this.front = this.rear = null;
    }
     public void enqueue(int key)
    {
          QNode temp = new QNode(key);
          if (this.rear == null) {
             this.front = this.rear = temp;
             return;
         }
          this.rear.next = temp;
         this.rear = temp;
    }
     public void dequeue()
    {         if (this.front == null)
             return;
          QNode temp = this.front;
```

```
        this.front = this.front.next;
         if (this.front == null)
            this.rear = null;
    }
}
```

Time Complexity: Time complexity of both operations enqueue() and dequeue() is O(1) as we only change few pointers in both operations. There is no loop in any of the operations.

Auxiliary Space: Space complexity of both operations enqueue() and dequeue() is O(1) as constant extra space is required.

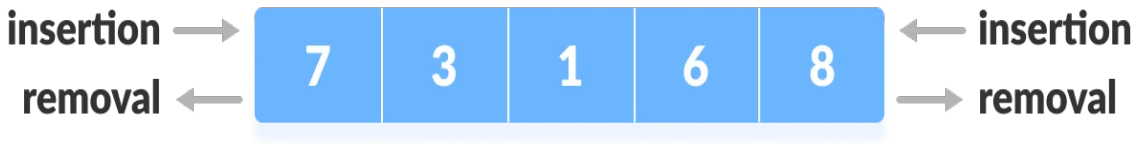
Deque

Deque is a type of queue in which insert and deletion can be performed from either **front** or **rear**. It does not follow the FIFO rule. It is also known as **double-ended queue**

Operations on Deque:

Deque consists of mainly the following operations:

- Insert Front
- Insert Rear
- Delete Front
- Delete Rear



1. Insert at the Front: This operation is used to add an element at the front. If the number of elements is not exceeding the size of the deque then only insertion takes place.

2. Insert at the Rear: If the deque is not full then this function inserts the element at the back end of the queue.

3. Delete from the Front: This operation is used to delete an element from the deque. If the deque is not empty then this operation will delete an element from the front end of the deque.

4. Delete from the Rear: This operation is used to delete an element from the back end of the deque if the deque is not empty.

Good implementation can be achieved using List<> property and classes are nearly identical to class that defines Queue. Deque can also be achieved combining two separate queue