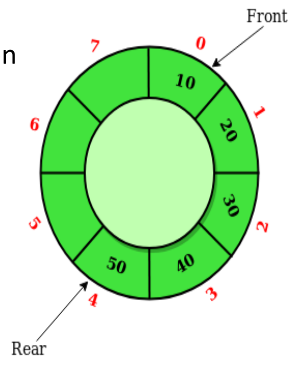
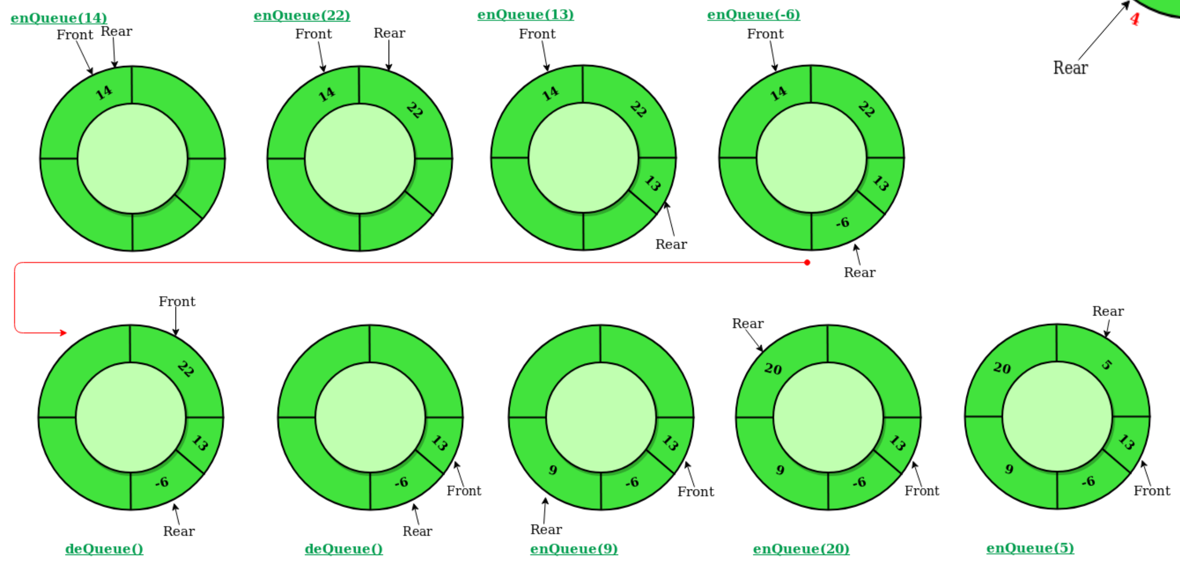
Circular Queue

You may read this article, I have taken material from this article: (https://www.geeksforgeeks.org/circular-queue-set-1-introduction-array-implementation/)



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 normal 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.



In queue implementation two pointers (not C++ pointers rather integer variables use to index array) front & rear are used. Variable **rear** is used to put elements on the back of the queue (conceptually) and **front** is used to get element from front of the queue (again conceptually), here I am using term conceptually refers to the real/ physical concept of the queue, whereas in circular queue implementation is quite different.

Operations on Circular Queue:

- **`enQueue(value)`** This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at **rear** position. The steps are:
  1. Check whether queue is Full:

     ```
     ( (rear == SIZE-1 && front == 0) || rear == front - 1 )
     ```

  - In first condition, if front is at 0 index (means no element is removed) and rear reaches to SIZE -1 means all the slots are captured and queue is full
  - In second condition (if some elements are already removed and front is not at 0), if rear reaches to the neighbor of front by moving clockwise into circle, then again all slots will be captured and queue will be full

  2. If queue is full then throw exception, because enqueuer operation is not possible. If queue is not full then, check if

     ```
     (rear == SIZE - 1 && front != 0)
     ```

  if it is true then set rear=0 and place element in the queue.

- **`deQueue()`** This function is used to delete an element from the circular queue. In a circular queue, the element is always deleted from front position. The steps are:

  1. Check whether queue is empty means check if

     ```
     (front==-1)
     ```

  2. If queue is empty, then display Queue is empty. If queue is not empty, then go to step 3

  3. Check if:

     ```
     (front==rear)
     ```

     if it is true then set:

     ```
     front = rear = -1
     ```

     else check if:

     ```
     (front==size-1)
     ```

     if it is true then set:

     ```
     front = 0
     ```

     and return the element.

Time Complexity: Time complexity of operations is O(1) as there is no loop in any of the operation.