Queue

ADT Queue

- What is a queue ?
  - A ***queue*** is a special kind of list in which … insertion is done on one side called **rear** … while deletion is done on the other side called **front**.
  - The queue is called **FIFO** (First In First Out) List.

chrisp 10

Operations: ENQUEUE, DEQUEUE, && FRONT

1. **ENQUEUE(x,Q)**
   - inserts the element x at the **rear** of the queue Q.
2. **DEQUEUE(Q)**
   - deletes the **front** element of the queue Q.
3. **FRONT(Q)**
   - returns the element at the **front** of the queue Q.

chrisp 12

Utility Operations:

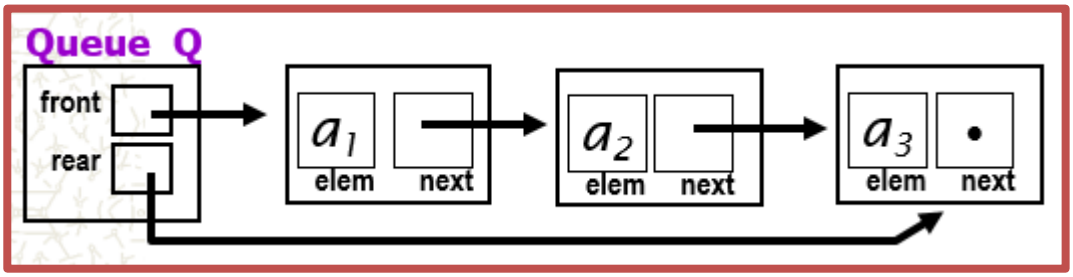
1. **initQueue(). Initializes the queue to be empty.**
2. **isEmpty(). Returns TRUE if the queue is empty; otherwise returns FALSE.**
3. **isFull(). Returns TRUE if the queue is full; otherwise returns FALSE.**

Implementations:

1) **Linked list Implementation**
2) **Circular Array Implementation**
3) **Cursor-base Implementation**

A) Linked List Implementation of Queue:

- **A queue is a structure containing front and rear pointers to dynamically allocated nodes containing the elements of the queue.**
- **Illustration:**



Practice Exercise 1: Linked List Implementation

1. **Write an appropriate definition of datatype Queue, as illustrated above.**
2. **Based on the definition in #1, write the code of the functions:**

   a) initQueue(). Given the queue, the function will initialize it to be empty, i.e. front and rear pointers will be NULL.

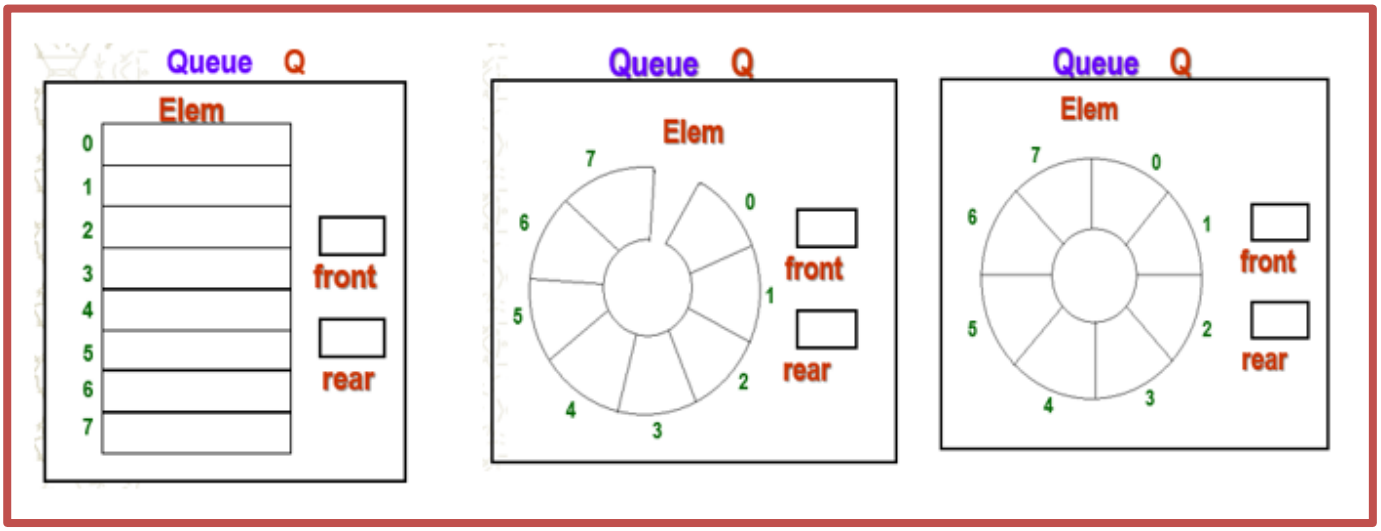
   b) enqueue(). Given the queue and an element, the function will insert the element at the rear of the queue. Note: Also consider the situation where the queue is initially empty and the first element is to be inserted.

   c) dequeue(). Given the queue, the function will delete the front element if it exists. Note: Also consider the situation where the queue has 1 element only, and deleting it will make the queue empty.

   d) front(). Given the queue, the function will return the front element if it exists. Note: Also consider the situation where the queue is empty.

B) Circular Array Implementation of Queue:

The queue is a structure containing an array of elements, and front and rear indices. Given below are 3 different conceptual views, but the same implementation.



Circular array implementation

- The array is declared in the same manner as any array in C
- Being circular is just an orientation, or a manner in which the array is manipulated or traversed.
- Like a circle, the array has no beginning and no end; hence the first element can be stored anywhere.
- Succeeding elements are inserted in CLOCKWISE or COUNTERCLOCKWISE direction; hence front and rear will also move clockwise or counterclockwise. Note: In the following operations, CLOCKWISE MOVEMENT will be used

- Caution!! To put MAX elements in the array of MAX size, will produce the following scenario:
  - 1 Element : Front is equal to Rear
  - Full Queue: Front is ahead of Rear by 1 cell
  - Empty Queue: Front is ahead of Rear by 1 cell

Problem: How do we distinguished an empty queue from a full queue?
Solution 1: Add a counter variable
Solution 2: Sacrifice 1 cell (Queue is considered full if there are (MAX – 1) elements, where MAX is the size of the array). Given below is the scenario for solution 2.

- 1 Element : Front is equal to Rear
- Full Queue: Front is ahead of Rear by 2 cells
- Empty Queue: Front is ahead of Rear by 1 cell

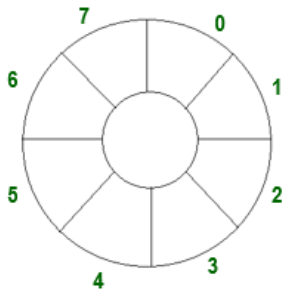
Practice Exercise 2: Circular Array Implementation, using solution 2.

1. **Write an appropriate definition of datatype Queue, as illustrated above. Include a macro definition of MAX, representing the size of the array.**

2. **Empty vs. Full Queue. Illustrated below are the pair of values for Rear and Front indices illustrating Empty and Full Queue.**

Front is ahead of Rear by 1 cell

Rear	Front
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	0



Front is ahead of Rear by 2 cells

Rear	Front
0	2
1	3
2	4
3	5
4	6
5	7
6	0
7	1

   a) **Using Rear and Front variables, write the condition that expresses that the queue is**
      i) **Empty**
      ii) **Full**

   b) **Using Rear OR Front variable, write the statement the will move Rear or Front to the next position.**

   Hint: Use modulo operator for a) and b)

3. **Based on the definition in #1 and condition and statement in #2, write the code of the functions:**

   a) **initQueue(). Given the queue, the function will initialize it to be empty, i.e. front and rear will be.**

   b) **enqueue(). Given the queue and an element, the function will insert the element at the rear of the queue. Note: Also consider the situation where the queue is initially empty and the first element is to be inserted.**

   c) **dequeue(). Given the queue, the function will delete the front element if it exists. Note: Also consider the situation where the queue has 1 element only, and deleting it will make the queue empty.**

   d) **front(). Given the queue, the function will return the front element if it exists. Note: Also consider the situation where the queue is empty.**