Assignment No. 13

Sumit Gulab Bhamare
SEComp A08
Sub - DSL

Aim : To illustrate the concept of circular queue

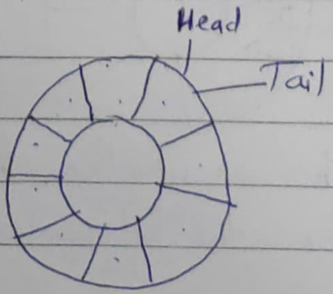
Problem Statement : Pizza parlor accepting maximum M orders. Orders are served in first come first served basis. Order once placed cannot be cancelled. Write C++ program to simulate the system using circular queue using array.

Learning Objectives : To understand concept of circular queue.
To analyze the various functions of circular queue.

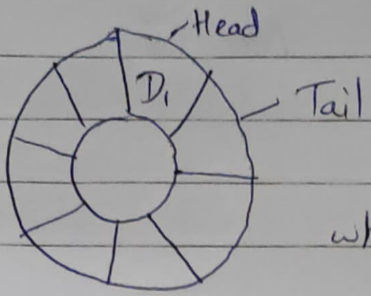
Learning Outcome : Students will be able to implement stack & queue data structures & algorithms for solving different kinds of problems.

Theory :
Circular Queue :
1. In case of a circular queue, head pointer will always point to the front of the queue, & tail pointer will always point to end of the queue.
2. Initially, the head & the tail pointers will be poiting to the same location, this would mean that the queue is empty.

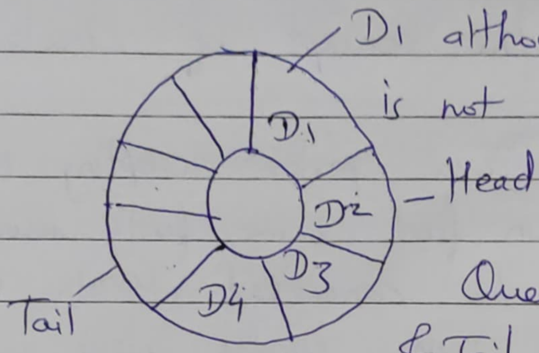
Initially the queue is empty, as Head & Tail are at same location.



A simple circular queue with size 8



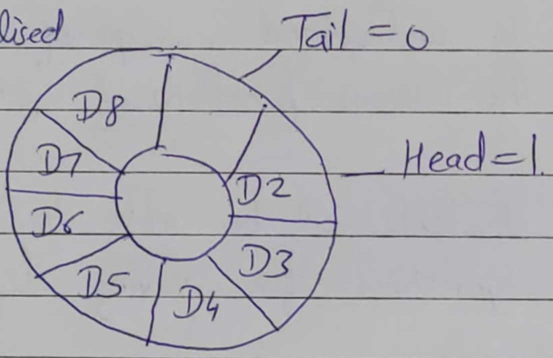
Tail always points to the location where new data will be inserted



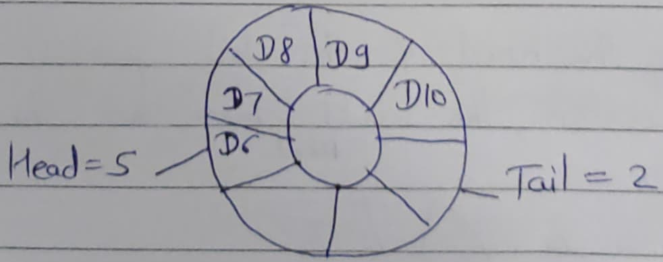
D1 although holds the same position but is not considered to be in the queue anymore.

Queue is only between Head & Tail, hence data in queue = D2, D3, D4.

Tail gets reinitialised to 0 after location 8, same will happen to the head.



Also the head & tail pointers can cross each other. In other words, head pointer can be greater than the tail. This will happen when we dequeue the queue a couple of times & the tail pointer gets reinitialized upon reaching the end of the queue.



In such a situation the value of the Head pointer will be greater than the Tail pointer.

Input : Enter the orders of Pizza Parlor

Output : Add orders & serve orders of pizza.

Algorithm : Implementation of circular queue

1. Initialize the queue, with size of the queue defined (maxSize), & head & tail pointers.

2. enqueue : Check if the number of elements is equal to maxSize -1 :
   - If Yes, then return Queue is full.
   - If No, then add the new data element to the location of tail pointer & increment the tail pointer.

3. dequeue : Check if the number of elements in the queue is zero :
   - If Yes, the return Queue is empty.
   - If No, then increment the head pointer.

4. Finding the size :
   - If, tail >= head, size = (tail - head) + 1
   - But if, head > tail, then size = maxSize - (head - tail) + 1

Software required : g++ / gcc compiler - / 64 bit fedora

Conclusion : Thus, we have studied the implementation of circular queue.