Assignment No. 11

Sumit Gulab Bhamare
SE Comp A 08
Sub - DSL

Aim: To illustrate concept of queue.

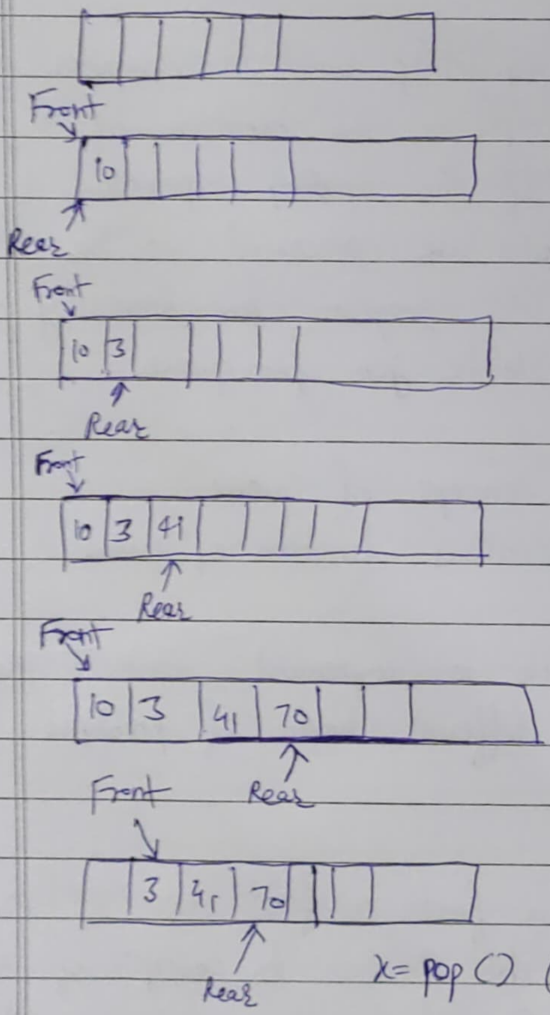
Problem Statement: Queues are frequently used in computer ~~engineering~~ programming, and a typical example is the creation of a job queue by an operating system. If the ~~opeating~~ operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue. Write functions to add job & delete job from queue

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

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

Theory:
A queue is logically a first in first out (FIFO or first come first serve) linear data structure. The concept of queue can be understood by our real life problems. For example a customer come & join in a queue to take the train ticket at the end (rear) & the ticket is issued from the front end of queue. That is, the customer who arrived first will recieve the ticket first.

Push operation will insert (or add) an element to queue, at the rear end, by incrementing the array index. Pop operation will delete (or remove) from the front end by decrementing the array index & will assign the deleted value to a variable. Total number of elements present in the queue is front - rear+1, when implemented using arrays.



Rear = -1
Front = -1

Rear = 0
Front = 0

Rear = 1   Front = 0

Rear = 2   Front = 0

Rear = 3   Front = 0

Rear = 3   Front = 1

x = pop() (i.e.; x = 10)

Input: Enter the jobs to an operating system.

Output: Add jobs & delete jobs from queue

Algorithm:

```
·  Algorithm to define class:
Step 1: class queue (element)
Step 2: declare create() A queue
Step 3: add (element, queue) A queue
Step 4: delete (queue) A queue
Step 6: get Front(queue) A queue
Step 7: Is_Empty(queue) A Boolean;
Step 8: For all Q belongs to queue, i belongs to element ~~list~~ let.
Step 9: Is_Empty(create()) = true
Step 10: Is_Empty(add(i,Q)) = false
Step 11: delete (create()) = error
Step 12: delete (add (i, Q)) =
                 if Is_Empty (Q)
                 then
                      create
                 else
                      add (i, delete (Q))
Step 13: get Front(create) = error
Step 14: get Front (add (i,Q)) =
                 if Is_Empty (Q)
                  then
                      i
                 else
                      get Front(Q)
                 end
Step 15: End.
```

· Algorithm to check whether the queue is empty or not

```
Step 1: Is_Empty()
Step 2: if (Front == Rear)
                return 1;
            else
                    return 0;
            End
Step 3: End.
```

· Algorithm to return the element at the front

```
Step 1: getFront()
Step 2: if(Is_Empty())
            then
                print Sorry, queue is Empty.
            else
                return (Queue[Front +1])
            End
Step 3: End
```

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

Conclusion: Thus we have studied the implementation of queue.