Queue →

Applications of Queue ⇒

- Booking of Tickets in Station.
- Aeroplane scheduling and Train Scheduling
- Getting milk from shop
- Getting Line at Toll tax
- Execution of programs
- Music Playlist
- Input output Buffer in CPU
- Event queues in Java scripts
- Allotment of Admission Nos
- Streaming & Downloading
- Video playing
- TV shows.
- Receiving/answering calls in call centre
- I/P - O/P buffer
- Youtube videos
- BFS in a graph
- Level order traversal in a tree
- Transitive closure of a graph

Queue → It is an ordered collection of item in which items may be inserted at the rear end of the queue and removed from another end called front end of queue.

Queue

```
ALGORITHM Initialize (Queue Q)
BEGIN:
        Q.FRONT = 0                 Time - O(1)
        Q.REAR = -1                 Space - O(1)
END;
```

```
ALGORITHM Empty (Queue Q)
BEGIN:
        IF Q.REAR - Q.FRONT + 1 == 0 THEN
                RETURN TRUE
        ELSE                                    T  O(1)
                RETURN FALSE                    S  O(1) No extra variable
END;
```

```
ALGORITHM EnQueue (Queue Q, x)
BEGIN:
        IF Q.REAR == SIZE-1 THEN
                WRITE ("Queue Overflow")        S - O(1) No extra variable
                EXIT(1)
        ELSE                                    T - O(1)
                Q.REAR = Q.REAR+1
                Q.Item[Q.REAR] = x
END;
ALGORITHM DeQueue (Queue Q)
BEGIN:
        IF Empty(Q) THEN                        S - O(1)
                WRITE ("Queue Underflows")      T - O(1)
                EXIT(1)
        ELSE
                x = Q.ITEM[Q.FRONT]
                Q.FRONT = Q.FRONT + 1
```

RETURN 7

END;

ALGORITHM TRAVERSAL (QUEUE Q)

BEGIN:

i = FRONT

WHILE (i <= REAR)

WRITE Q(i)

i++

END;

ALGORITHM TRAVERSAL (QUEUE Q)

BEGIN:

i = FRONT

WHILE (i <= REAR)

WRITE (Q(i))

i++;

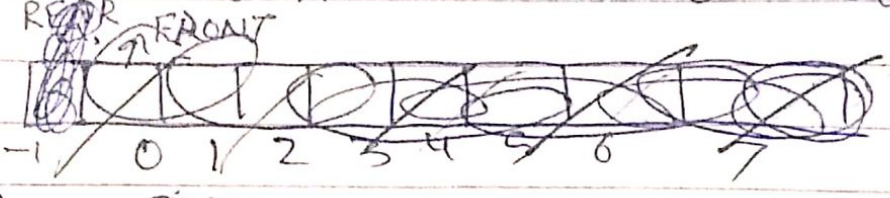
END;

FOR (i = 0 FRONT TO

WRITE (

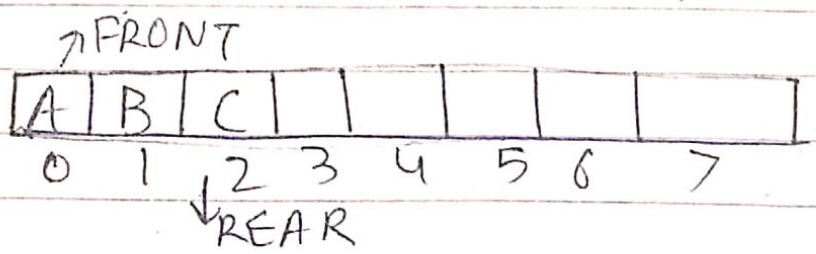
Given an empty Queue of size 8. Show the result of following operations on Queue

3 items A, B, C are inserted
1 item is deleted.
3 data items D, E, F are inserted
2 items are deleted
3 data items X Y Z are inserted



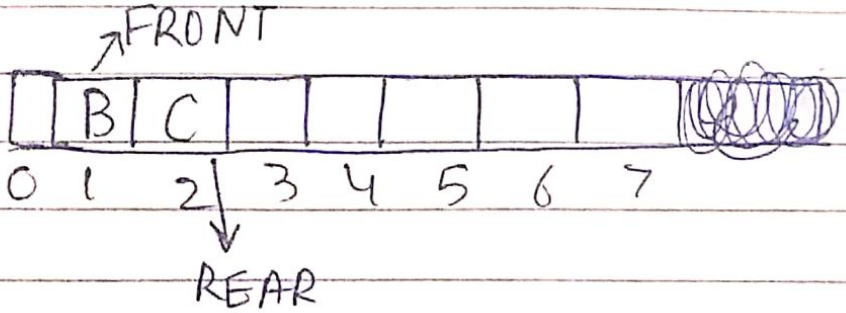
FRONT = 0
REAR = -1

(1)

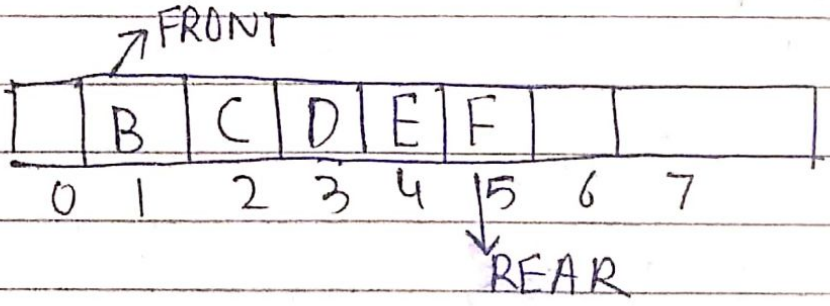


Deque of Queue

(2)



x = A



x = B

x = C

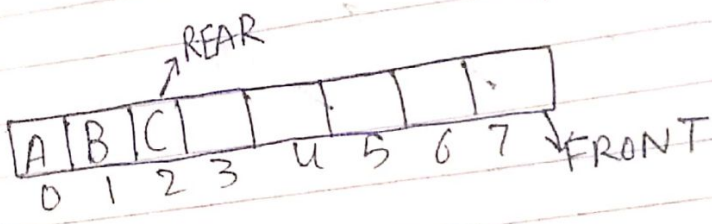
WHILE insertion of Z overflow occurs

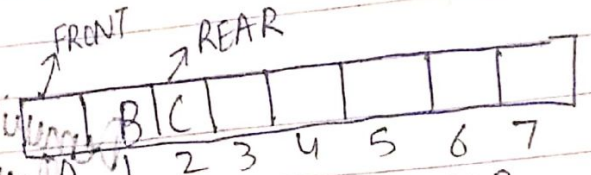
If rear position has reached to last position of the array and insertion is attempted even if we have empty cell towards beginning, then the attempted insert will result in overflow condition.

Circular Queue ⇒

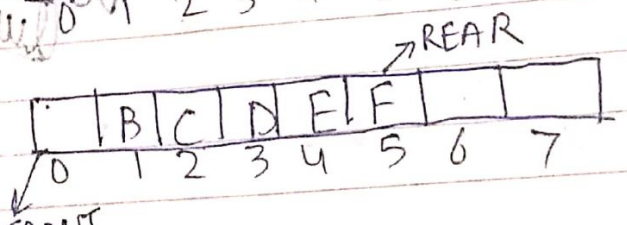
FRONT = 7
REAR = 7

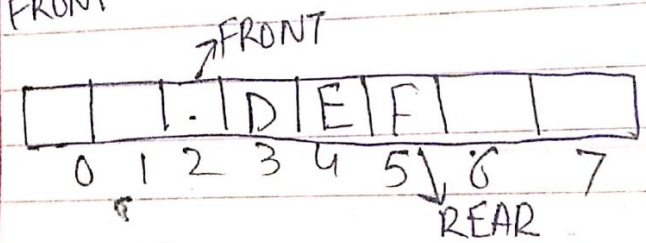
(r+1)%8

① 

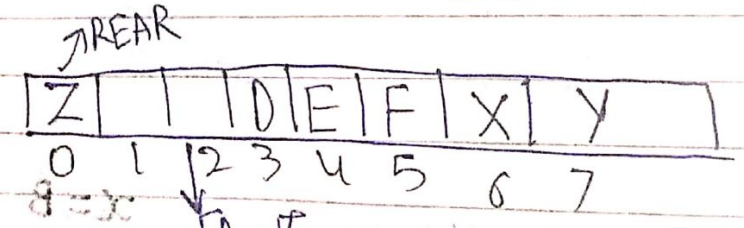
② 

x = A

③ 

④ 

x = B
x = C

⑤ 

(r+1)%8

Rear = 0
Front = 2

```
ALGORITHM Initialize (Queue CQ)
BEGIN:
        CQ.REAR = SIZE - 1          T(Q1)
        CQ.FRONT = SIZE - 1         S(Q1)
END;


ALGORITHM Empty (Queue CQ)
BEGIN
    IF CQ.REAR == CQ.FRONT THEN
            RETURN TRUE
    ELSE                                T(Q1)
            RETURN FALSE                S(Q1)
END;


ALGORITHM DeQueue (CQueue CQ)
BEGIN:
        IF Empty (CQ) THEN
            WRITE ("Queue Underflow")       T(Q1)
            EXIT(1)                         S(Q1)
        ELSE
            CQ.FRONT = (CQ.FRONT + 1) % Size

            x = (Q.Item[CQ.Front])
            RETURN x
END;
```

ALGORITHM ENQUEUE (CQueue CQ , x)
BEGIN :

IF ((CQ.REAR + 1) % SIZE == CQ.FRONT
WRITE (" Queue Overflow ")
EXIT(1)

CQ.REAR = (CQ.REAR + 1) % SIZE
CQ.Item [CQ.REAR] = x

END ;

T(O, S(O,

ALGORITHM Traverse (CQueue CQ)
BEGIN :

~~i = CQ.FRONT+1, (i % SIZE) != CQ.REAR~~

FOR (i = CQ.FRONT TO CQ.REAR) DO
WRITE (CQ.Item [i])

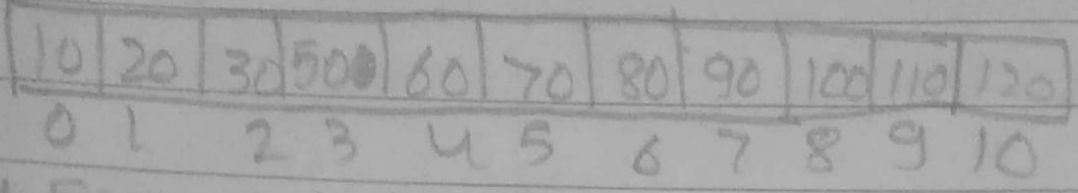
ALGORITHM Traverse (CQueue CQ)
BEGIN :

i = (Q.FRONT + 1) % SIZE

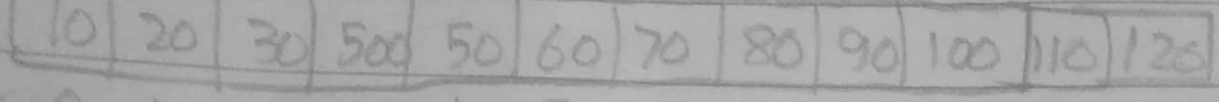
WHILE i % SIZE != Q.REAR DO
WRITE (Q.ITEM [i])
i++

WRITE (CQ.Item [i % SIZE])

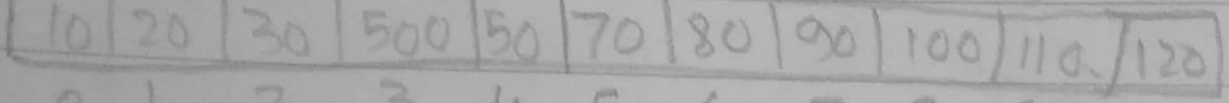
Scanned with CamScanner



ed 500 at index 3



lution ~~at~~ index 5 :



Priority Queue

ALGORITHM ArrayInsertion(A[], N, i, x)
BEGIN;

N ↕ FOR j = N-1 TO i STEP-1 DO
A[j+1] = A[j]

2 ↕ A[i] = x
N = N+1

END;

ALGORITHM ArrayDeletion(A[], N, i)
BEGIN;
x = A[i]
FOR j = i+1 TO N-1 DO
A[j-1] = A[j]
N = N-1
RETURN x
END;