ADITYA RAJ   B100
U20CS100
DS TUTORIAL - 6

1.a.

QINSERT(QUEUE, N, FRONT, REAR, ITEM)

This algorithm enters an element ITEM into a queue.

```
1. START.
2. If REAR = N-1
        Then write: Overflow and return.
3. Else if FRONT = N or if REAR =
        -1 Then Set FRONT ->
        REAR -> 0
4. Else
        Set REAR -> REAR+1.
   [End of if structure]
5. Set QUEUE[REAR] -> ITEM.
6. EXIT.
```

1.b.

QDELETE(QUEUE, N, FRONT, REAR, ITEM)

This algorithm deletes an element from a queue and assigns it to the variable ITEM.

```
1. START.
2. Set ITEM = QUEUE[FRONT].
3. If FRONT = N or if REAR = -1
        Then write Underflow and Return.
4. Else
        Set FRONT -> FRONT+1.
   [End of if structure]
5. EXIT.
```

1.c.

QFRONT(QUEUE, N, FRONT, REAR)

This algorithm returns the value at front or peek of the QUEUE.

```
1. START.
2. If FRONT = N or if REAR = -1
```

```
        Then write EMPTY and Return.
3. Else
        Write QUEUE[FRONT].
   [End of if structure]
4. EXIT.
```

1.d.

isFull(QUEUE, N, FRONT, REAR)

This algorithm returns bool whether QUEUE is full or not.

```
1. START.
2. If REAR = N-1
        Then return TRUE.
3. Return FALSE.
```

1.e.

isEmpty(QUEUE, N, FRONT, REAR)

This algorithm returns Boolean whether QUEUE is empty or not.

```
1. START.
2. If REAR = -1 or if FRONT = N
        Then return TRUE.
3. Return FALSE.
```

2.a.

REARInsert(DEQueue, N, FRONT, REAR, ITEM)

This algorithm insert an element ITEM at the rear of the double ended queue.

```
1. START.
2. If REAR = N-1
        Then write Overflow and return.
3. Else if REAR=FRONT=-1
        Then set FRONT -> FRONT+1
        Set REAR -> REAR+1.
4. Els
   e    Set REAR -> REAR+1.
   [End of if structure]
5. Set DEQueue[REAR] -> ITEM.
6. EXIT.
```

2.b.

FRONTInsert(DEQueue, N, FRONT, REAR, ITEM)

This algorithm insert an element ITEM at the front of the double ended queue.

```
1. START.
2. If FRONT = 0
        Then write Overflow and return.
3. Else if REAR=FRONT=-1
        Then set FRONT ->
            FRONT+1 Set REAR -
            > REAR+1.
4. Else
        Set FRONT -> FRONT-1.
   [End of if structure]
5. Set DEQueue[FRONT] -> ITEM.
6. EXIT.
```

2.c.

FRONTdelete(DEQueue, N, FRONT, REAR)

This algorithm deletes the element at the front of the double ended queue.

```
1. START.
2. If REAR=FRONT=-1
        Then write underflow and return.
3. Else if REAR=FRONT
        Then set REAR=FRONT=-1.
4. Else
        Set FRONT->FRONT+1.
```

```
5. EXIT.
```

2.d.

REARdelete(DEQueue, N, FRONT, REAR)

This algorithm deletes the element at the rear of the double ended queue.

```
1. START.
2. If REAR=FRONT=-1
        Then write underflow and return.
3. Else if REAR=FRONT
        Then set REAR=FRONT=-1.
4. Else
        Set REAR->REAR-1.
5. EXI
   T.
```