1] Write Algorithm to add element into the Queue

Procedure QUEUE_ADD (A, Front, rear, size, ele)

Description :-

This procedure add new element in a simple queue. 'A' is a linear array A(1:size), 'Front' is a pointer pointing to front element in the queue. 'rear' is a pointer pointing to last element in the Queue. 'Size' is maximum capacity of Queue and 'ele' is the new element to be added in queue.

Declaration :-

```
        Global integer A(1:size),
                int front, rear, size
        parameter int ele.
```

Algorithm :-

```
        if rear = size, then
            print ("Queue is full")
        endif
        if front ← 0, then
            f ← 1
        endif
        rear ← rear + 1
            A(rear) ← ele
END QUEUE_ADD
```

2] Write algorithm to delete an element from the queue.

Procedure Queue_DEL (A, front, rear, size)

Description :-

This procedure delete an element from queue. A queue is maintain using array 'A' (1: size), where 'size' is maximum capacity of queue. 'front' and 'rear' are the pointers pointing to first and last element in queue. respectivly.

Declaration :-

Global integer : int A, front, rear
parameter int ele.

Algorithm :-

```
if front = 0, then
    return NULL
end if
    ele <- A(front)
if front = rear, then
    front <- rear <- 0
else
    front <- front +1
end if
return (ele)
```

END QUEUE_DEL

3] Write Algorithms to LIST_ALL elements present in circular Queue.

Procedure QUEUE-LIST_ALL (A, size, front, rear)

Description:-

This procedure shows list of all elements presents in circular Queue. 'A' is a linear array A(1:size), 'size' is maximum capacity of Queue, 'front' and 'rear' are pointers pointing to first and last element in the circular queue.

Declaration:-

Global - int A(1:size),
int front, rear, size.

Algorithm:-

3

```
if front = 0, then
    print ("Queue is empty")
else
    if front ≤ rear, then
        for i ← front to rear do
            print (A(i))
        repeat
    else
        for i ← front to size do
            print (A(i))
        repeat
        for i ← 1 to rear do
            print (A(i))
        repeat
endif
```

```
endif
END_QUEUE_LIST_ALL
```

4

Queue using Linked list :-

1) write a Program to add eelement in Queue.

Procedure ADD_Queue(front, rear, data, next, ele)

Description :- This procedure add a new node in queue organized using linked list.

'front & rear' are pointers to printing two nodes at begining and end of queue which are initially set to Null, when queue is empty.

'data' is a part of node that holds address of next node to from a link.

'ele' is an element which is add to be in queue.

Declaration :-

Global integer front, rear
parameter int ele

5

Algorithm :-

```
        if AVAIL = NULL, then
            print ("Queue is full")
        else
            NEW ← DEL (AVAIL)
            NEW → data ← ele
            NEW → next ← ze
            if front = NULL, then
                front ← NEW
                rear ← NEW
            else
                rear → next ← NEW
                rear ↔ NEW
            endif
    endif
END ADD_QUEUE
```

te for :
nm
Chart
amme Listing
lts
ments

2) corite a program for delete the queue.

Procedure DEL_QUEUE (front, rear, data, next)

Description :- This procedure detete ~~the onde~~ a node from queue orgranized using linked list.

'front' & 'rear' are pointers to pointing two nodes at begining and end of queue which are initially set to NULL, when Queue is empty.

'data' is a part of node that holds information & 'next' is a part of node that holds address of next node to from a link.

'AVAIL' is a list of free nodes.

Declaration :-

Global pointer start
int data, pointer start
parameter int ele.

Algorithm :-

```
if front = x, then
    print ("Queue is empty")
    return NULL
else
    ele ← front → data
    Temp ← front
    if front = rear, then
        rear ← NULL
    endif
    front ← front → Next
    ADD (AVAIL) ← Temp
    return (ele)
endif
```

★ Circular Queue:-

1] Write Algorithm to add new element in a cricular queue.

Procedure QUEUE_ADD (A, size, front, rear, ele)

Description:-

This procedure add new element in a queue using array in circular passion. 'A' is a linear array A(1:size). 'front' is a pointer pointing to front element in the queue. 'rear' is a pointer pointing to last element in the queue. 'size' is maximum capacity of Queue and 'ele' is the new element to be added in queue.

Declaration:-

```
        Global - A(1:size)
                 int front, rear, size
        parameter :- int ele.
```

Algorithm:-

```
    if front = 1 and rear = s or r+1 = f, then
        print (" Queue is full")
    else
        if f = 0
            f <- f+1
        endif
        if rear = size, then
            rear = 0
            endif
                r<- r+1
                A(r) <- ele
        endif
   END QUEUE_ADD
```

2] Write Algorithm to delete an element from circular Queue.

Procedure QUEUE_DEL (A, Size, front, rear)

Description :-

This procedure delete an element in a circular queue. 'A' is a linear array A(1:size), 'front' & 'rear' are the pointers pointing to the first and last element in the queue. 'size' is a maximum capacity of Queue.

Declaration :-

Global : int A (1:size), int front, rear, size

Algorithm :-

```
            if front = 0, then
                print ("Queue is empty")
            else
                ele <- A(front)
                if front = rear, then
                    front <- rear = 0
                else
                    if front = size, then
                        front <- 0
                    endif
                    F <- F+1
                endif
                return (ele)
            endif
   END QUEUE_DEL
```