* Outcome
1) Result of addition of job operation on queue
2) Result of deletion of job operation on queue

* Theory
* Queue
A queue is a linear data structure that follows the first out (FIFO) priciple. This means that the first element added to the queue is the first one to be removed. Queues are commonly used in scentarious where ordering is important

Types of Queues

1) Simple Queue
A basic fifo structure

2) Circular Queue
The last position is connected back to the first position making back to the back first position in specifitent in space utilization

3) priority Queue
element are dequed based on priority rather than order of insertion

4) Double - ended Queue
element can be added or removed from both end

* Advantage
order preservation : maintains the order of element ensuring that they are processed in the same sequence they were added

* Disadvantage.
overhead in dynamic implementations if implemented using linked lists each element requires additional memory for pointers

* Queue as an Abstruct Data Type (ADT)
A queue as an ADT defines a collection of operations that can be performed on a queue without specifying it's internet structure. This abstraction allow for flexibility in implementation

Basic operation of Queue ADT:-

1) enqueue (element): Adds an elements to the rear of the queue

```
procedure enqueue (element):
    If Is full ():
print "Queue overflow"
    Return
```

4) Is empty(): check if queue is empty
function Is empty():
Return front == -1

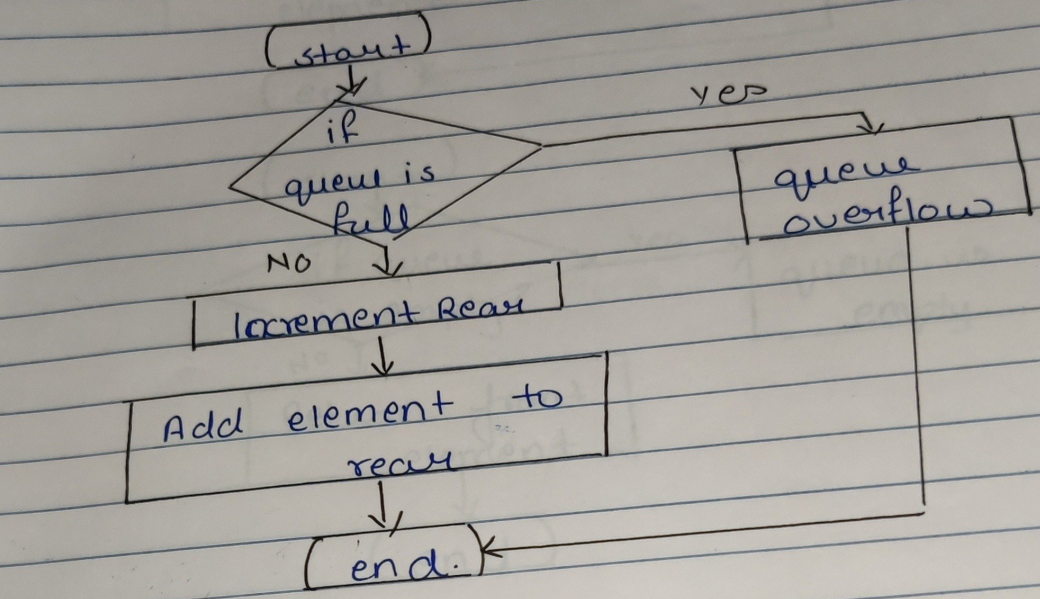
5) Is full() : check if the queue is full (if applicable)
function Is full():
Return rear == max - size - 1

* Algorithm

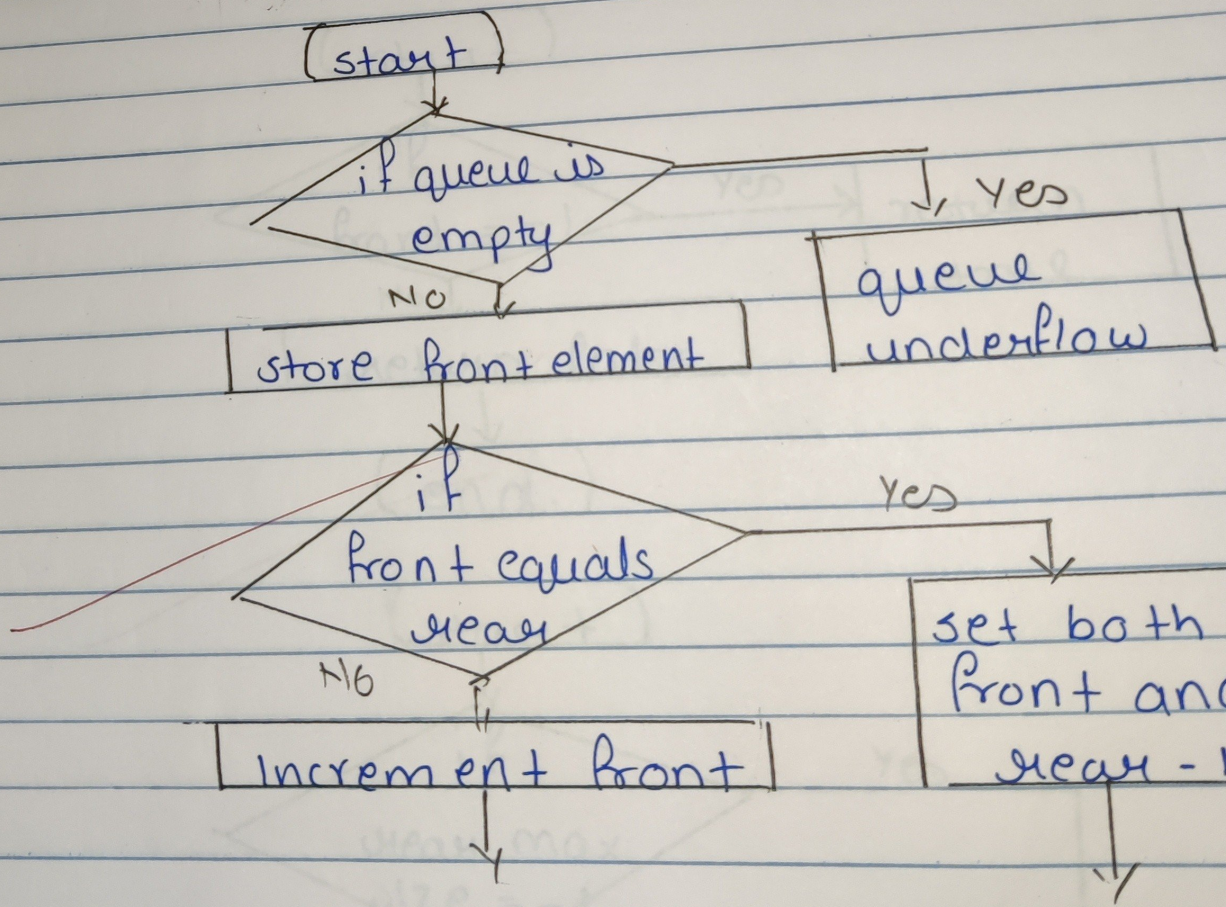
1) enqueue operation (Adding an element)
step 1) start
step 2) Check if the queue is full:
If it is full display "Queue overflow" and stop
step 3) Check if the queue is empty
step 4) Increase the rear index by 1
step 5) place the new element at the position of the rear index in the queue
step 6) The element has been added
step 7) stop

2) Dequeue operation (Removing an element)
step 1) start
step 2) check if the queue is empty
step 3) store the element at the front index in a variable to return later
step 4) check if the first and rare indexs quesame
step 5) If the queue is not empty increases front index by 1

* flowchart



2)



Return stored element

↓

(end)

3)

(start)

↓

if queue is empty? —Yes→ queue is empty

↓ No

Return front element

↓

(end)

4)

(start)

↓

if front = -1 —Yes→ return true

↓ No

return false

↓

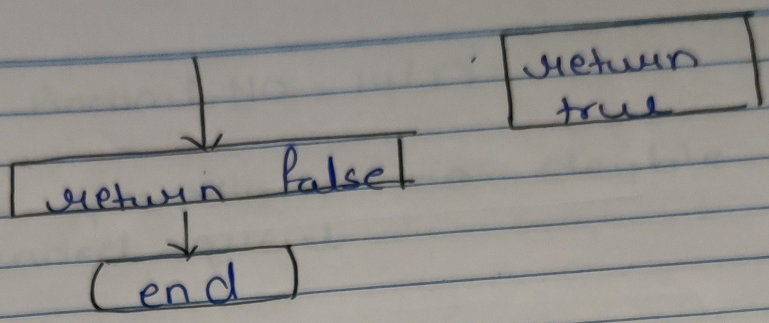
(end.)

(start)

↓

if rear max size = -1 —Yes→

↓ No



* Conclusion

By this way, we can perform operation on the double ended and different operation on queue.