14/9/19

CH 4 : Queue

↝ It is a linear DS in which the elements that is inserted first is the one that will come out first.

↝ It works in FIFO (first in first out) manner.

* Types Of Queue

→ Simple Queue

→ Circular Queue

→ DeQueue [Double Ended Queue]

→ Priority Queue

↝ Queue can be implemented using array & linked list.

* Basic Operation On Queue

→ enqueue ( ) – for insertion
→ dequeue ( ) – for deletion

QINSERT (Q, F, R, N, Y)

- Q – Name of Queue
- F – Front
- R – Rear
- N – Size of Queue
- Y – Element to be inserted

S.1 [Check for Overflow]
if R ≥ N
then write ("Overflow")
Exit ( )

S. 2 [Increment Rear pointer]
R ← R+1

S.3 [Insert Element]
Q[R] ← Y

S.4 [Is Front Pointer Properly Set?]
if F = 0
then F ← 1
Return

QDELETE (Q, F, R)

S.1 [Underflow ?]

If F = 0

then write ("underflow")

Return (0)

S.2 [Delete Element]

Y ← Q[F]

S.3 [Queue Empty ?]

if F = R

then F ← R ← 0

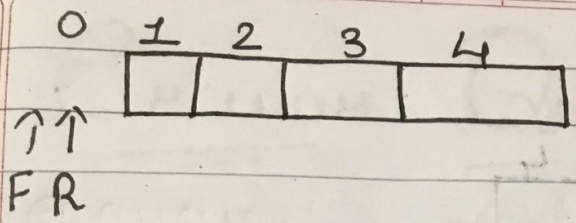
Else

F ← F + 1

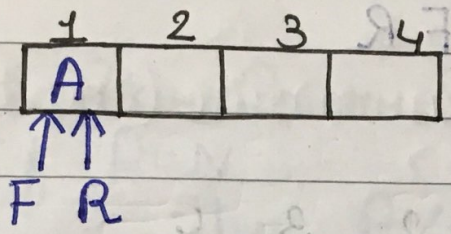
S.4 [Return Element]

Return (Y)

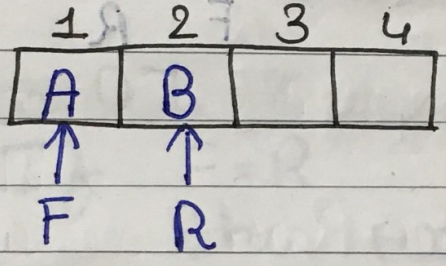
Ex:



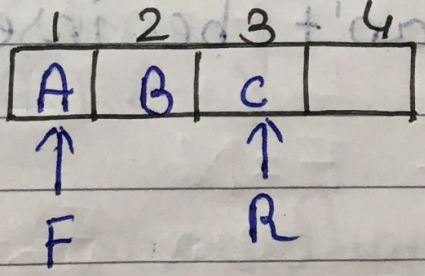
(i) Insert A



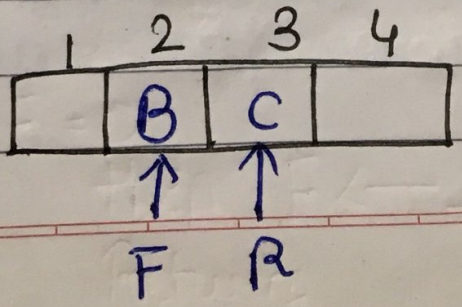
(ii) Insert B



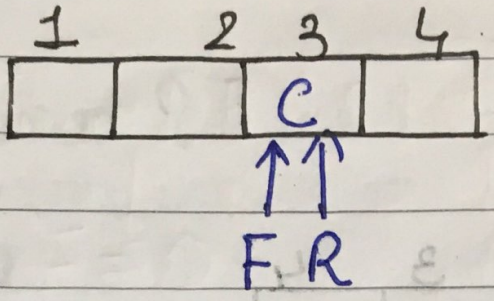
(iii) Insert C



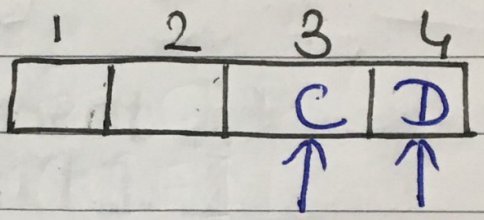
(iv) Delete



(v) Delete



(vi) Insert D



(vii) Insert E

R ≥ N, therefore it will be overflow

∴ E can't be inserted

* Circular Queue

CQINSERT (Q, F, R, N, Y)

S.1 [Reset Rear Pointer ?]
If R=N
then R ← 1
Else
R ← R+1

S.2 [Overflow ?]
If F=R
then write ("Overflow")
return

S.3 [Insert Element]
Q[R] ← Y

S.4 [Is Front Pointer Properly Set ?]
if F=0
then F ← 1
return

CQDELETE (Q, F, R, N)

S.1 [underflow ?]

If F = 0

then write ('underflow').

return (0)

S.2 [Delete Element]

Y ← Q[F]

S.3 [Queue Empty ?]

If F = R

then F ← R ← 0

else return (Y)

S.4 [Increment front Pointer]

If F = N

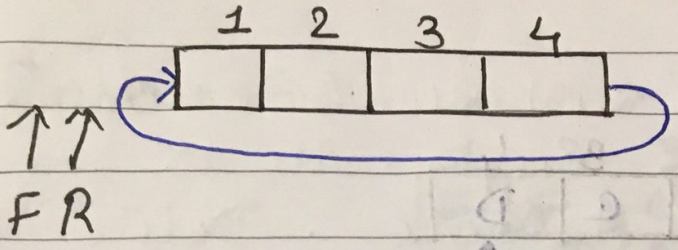
then F ← 1

Else

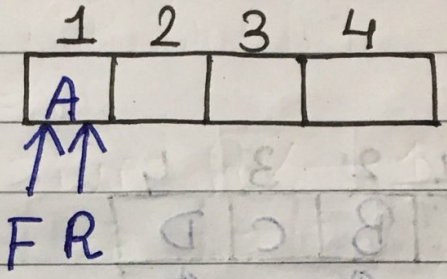
F ← F + 1

return (Y)

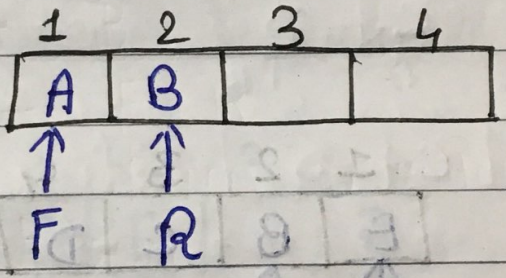
Ex:



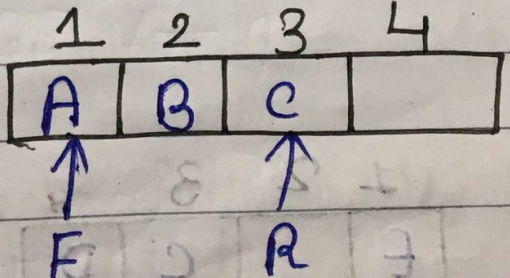
(i) Insert A



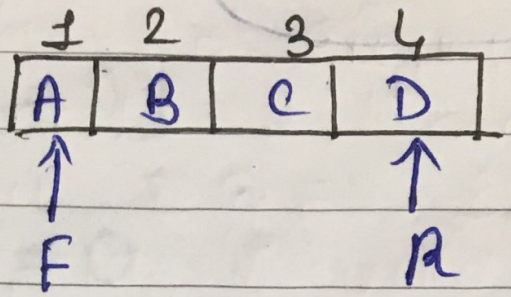
(ii) Insert B



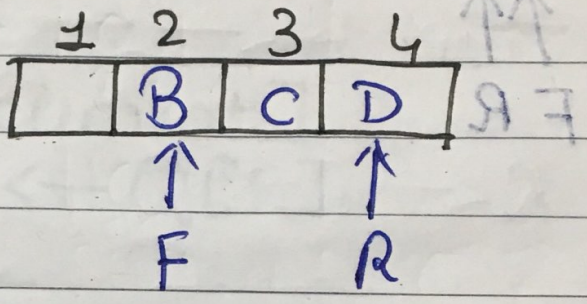
(iii) Insert C



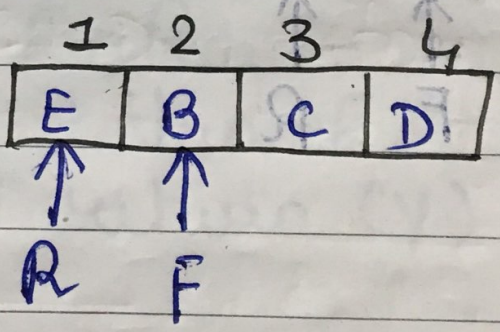
(iv) Insert D



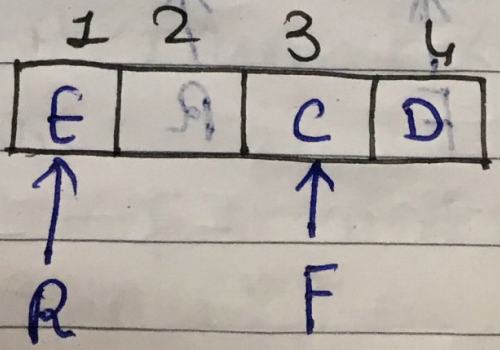
(v) Delete



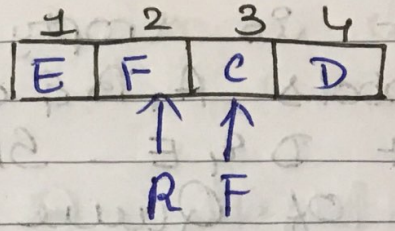
(vi) Insert E



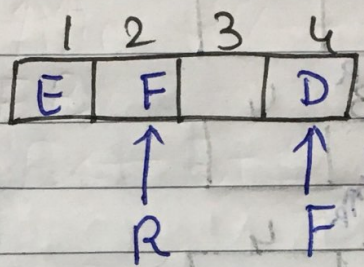
(vii) Delete



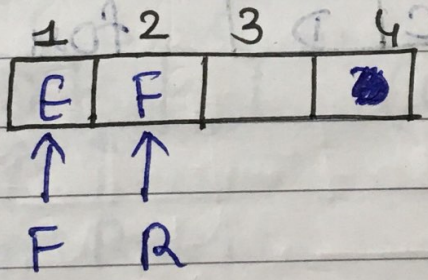
(viii) Insert F



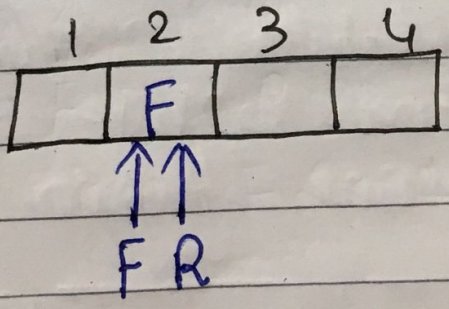
(ix) Delete



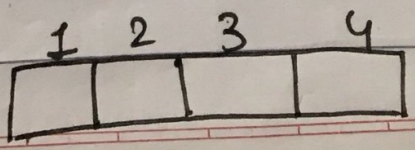
(x) Delete



(xi) Delete



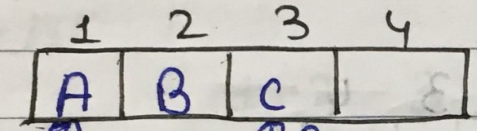
(xii) Delete



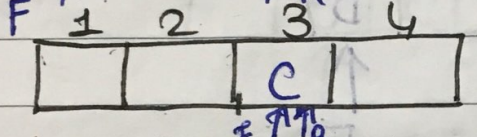
F = R = 0

Ex: Consider an Ex where Size of Queue 4. Initially it is empty. It is req. to insert symbols A, B & C. Delete A & B And Insert D & E. Show the trace of content of Queue

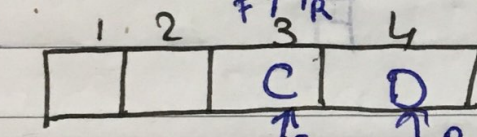
1.



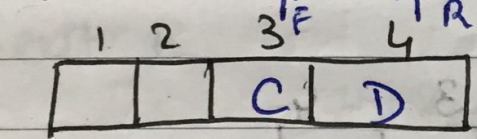
2.



3.



4.



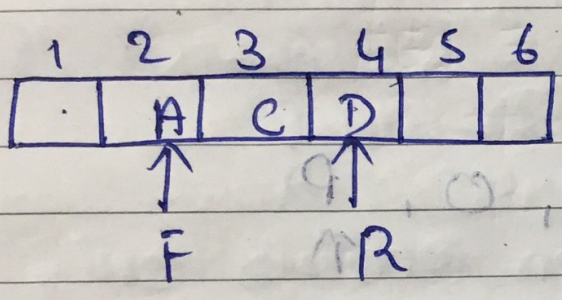
for E it is Over-flow

Ex: Consider foll. Queue where Q is CQ having 6 memory cell. F=2, R=4

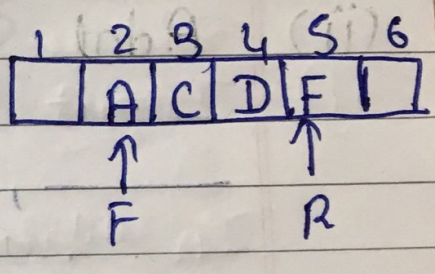
Queue: ___, A, C, D, ___, ___

(i) F added to Q (ii) 2 are deleted
(iii) R " " " (iv) S added to Q
(v) Delete

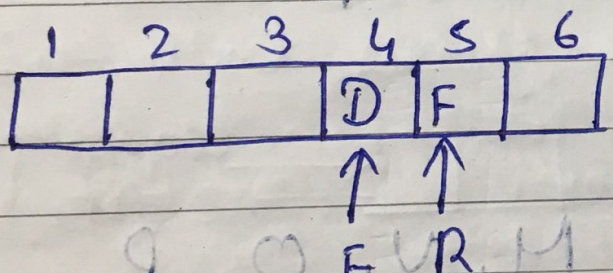
(1)



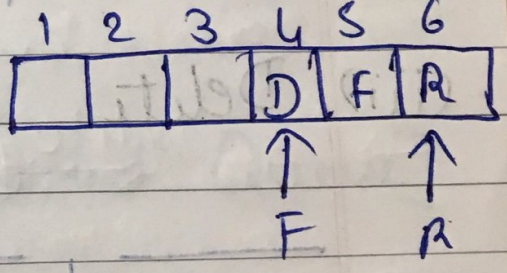
(2)



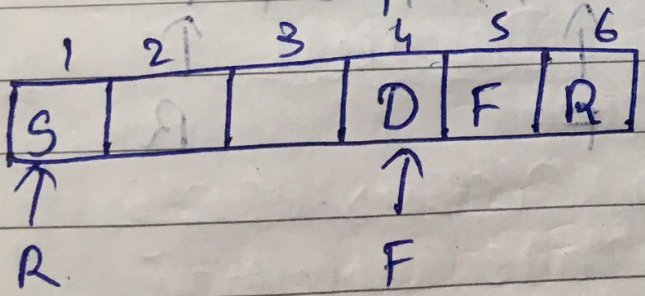
(3)



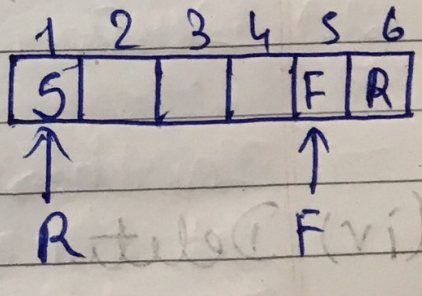
(4)



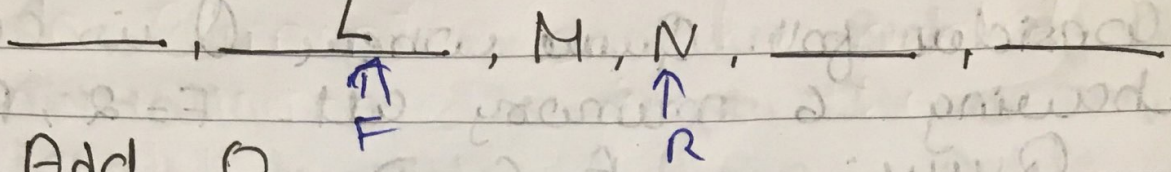
(5)



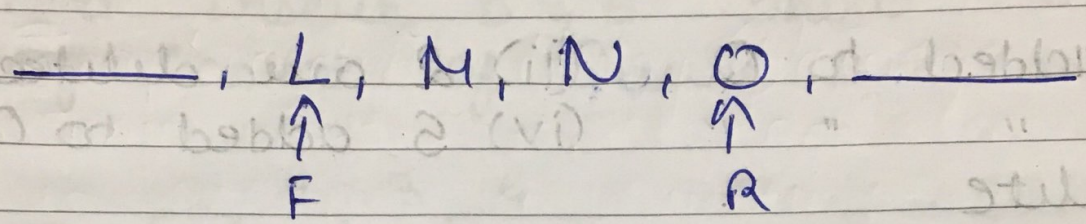
(6)



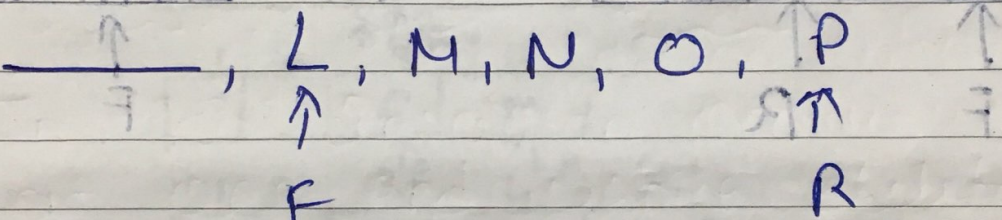
Ex:



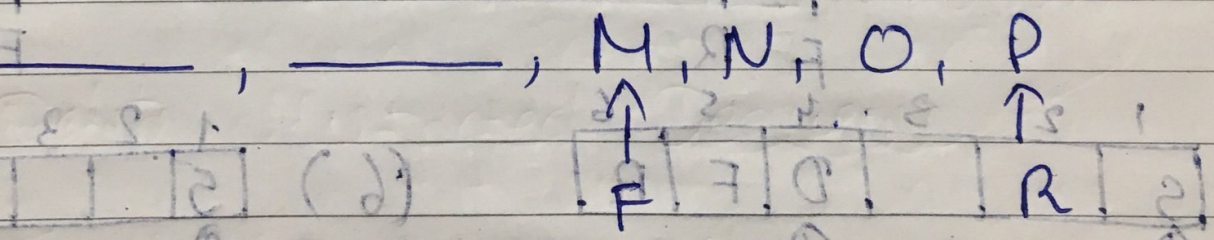
(i) Add O



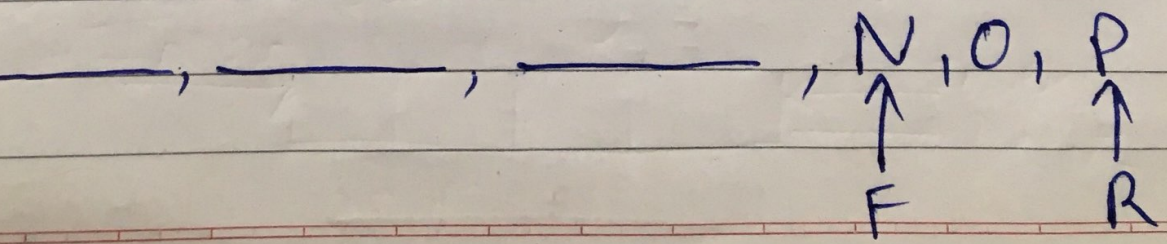
(ii) Add P



(iii) Delete



(iv) Delete



(vi) Add Q, R, S

Q , R , S , N , O , P

R F

(vii) Delete

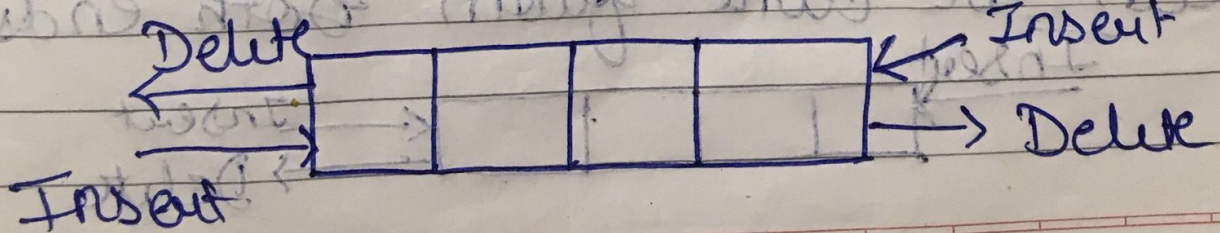
Q, R, S, ——— , O , P

R F

17/9/19

* DeQuue [Double Ended Quue]

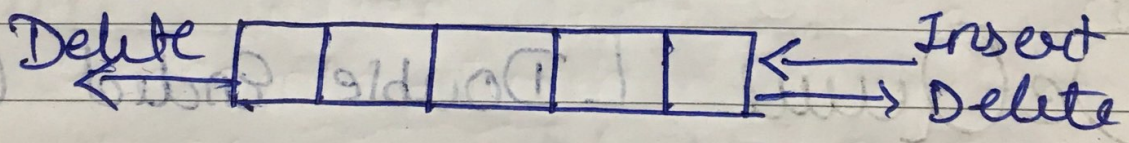
(a) DeQuue is a list in which elements can be inserted or deleted at either end.



- No Elements can be added or removed from the middle.
- Implemented using Doubly linked list.
- Two Variation of dequeue

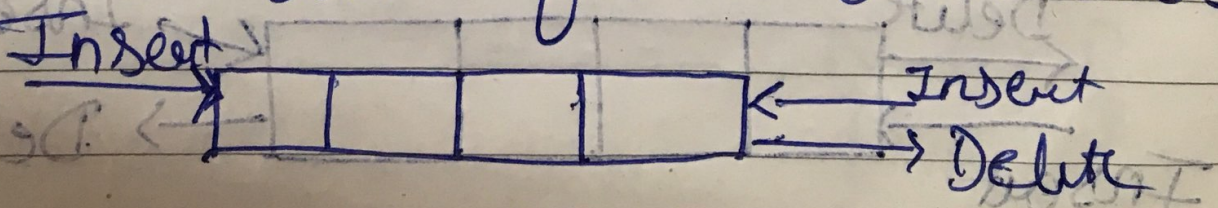
① Input Restricted Dequeue

In this insertion occurs only at one end while deletion can be done from both ends.



② Delete Restricted Dequeue

In this Deletion occurs only at one end while insertion can be done from both ends.



Ex: —, A, B, C, D, E, —, —, —, —

(i) Add f on left

F, A, B, C, D, E, —, —, —, —

(ii) Add G on right

F, A, B, C, D, E, G, —, —, —

(iii) Add H on right

F, A, B, C, D, E, G, H, —, —

(iv) Delete 2 alphabets from left

—, —, B, C, D, E, G, H, —, —

(v) Add I on right

—, —, B, C, D, E, G, H, I, —

* Priority Queue

Priority Queue is Data Structure in which each element is assigned a priority.

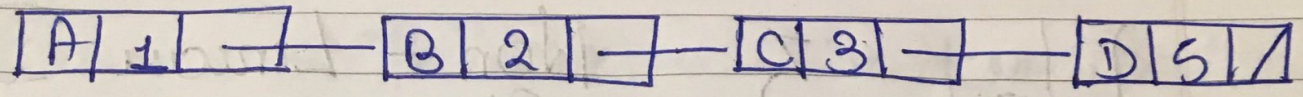
This priority of elements will be used to determine the order in which the Elements will be proced.

Rules for Priority Queue

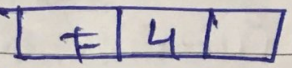
→ An element with higher Priority is proced before an element with lower Priority.

→ Two Elements with same priority are proced based on FCFS.

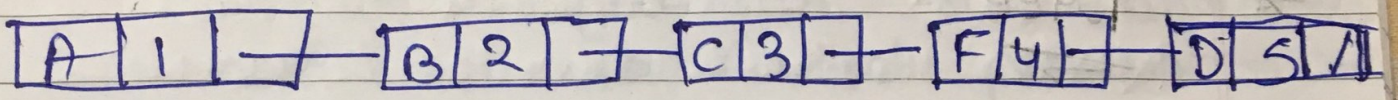
Initially



new Process



Now



* Application Of Queue

1. Job Scheduling
2. Categorizing Data
4. Josephus Problem