

Stack: ①

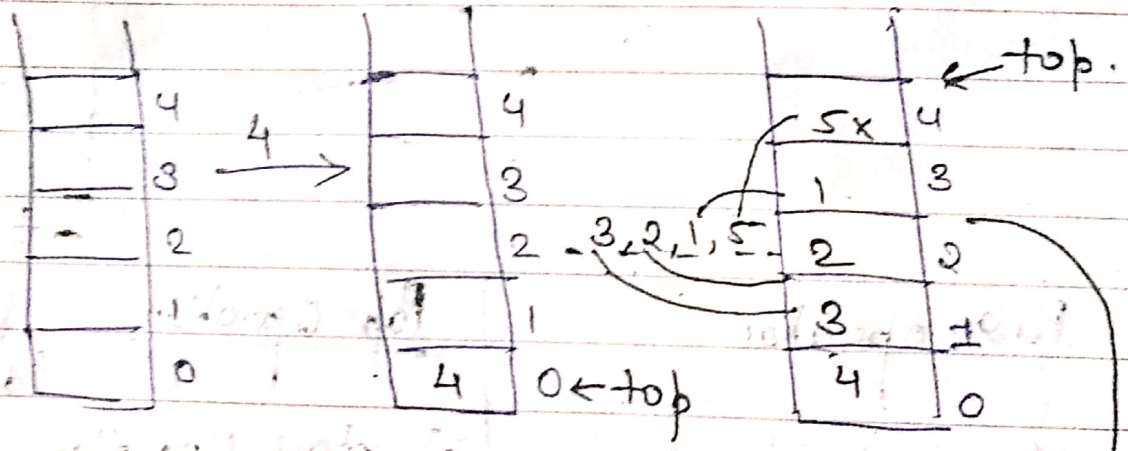
A ds in which last item inserted is taken out 1st.

(LIFO) - Last in 1st out.

Insertion

- One item is inserted / PUSH at a time on top of stack.
- Only one item is deleted / POP at a time from top of stack.

Operation $\begin{cases} \text{Push} \\ \text{Pop} \end{cases}$



When stack is empty = $\text{top} - 1$.

1 to insert = 1. have to insert 4 at the top of the stack.

Push = $\text{top} = \text{top} + 1$.

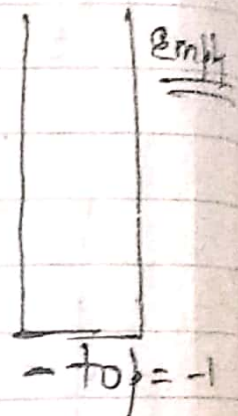
Deletion

(2)

5x	4	top
1	3	
2	2	
3	1	
4	0	

1	3 ← top
2	
3	
4	

$$\text{Pop} = \text{top} = \text{top} - 1$$



Push operation

1) Stack Overflow?

Stack is full

2) if $\text{Top} = \text{MAX} - \text{Stack}$
write overflow & exit

3) Read item (4)

4) Set $\text{Top} = \text{Top} + 1$ ($1 + 1 = 2$)

5) Set $\text{Stack}[\text{Top}] = \text{Item}$

6) Exit

Pop Operation

(delete)

1) Stack Underflow

↓ Stack is empty

if $\text{Top} = -1$ then
write underflow & Exit.

2) Repeat step 3 to 5 until

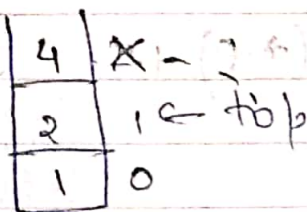
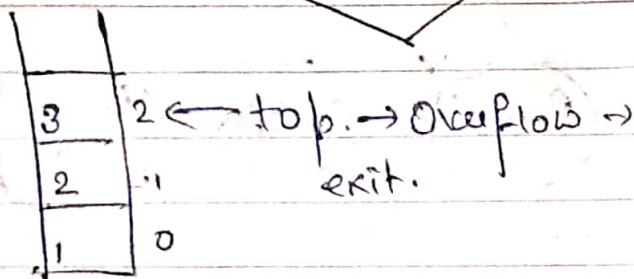
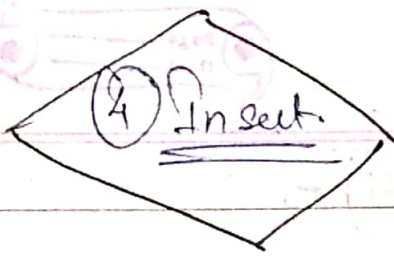
$\text{Top} \geq 0$

3) Set $\text{Item} = \text{Stack}[\text{Top}]$

Item = Stack[2]

Item = 4, (2), 1

3

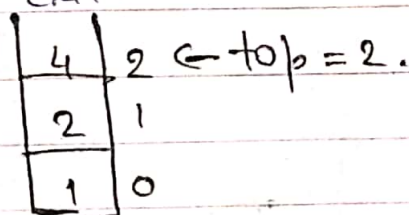


deleted item 4.

(4.) Set $Top = Top - 1$.

(5.) Delete deleted item = 4

(6.) Exit.



$top = -1$, 0, -1
deleted item 4, 2, 1

Example of Infix to postfix conversion:-

(1) $(A * B) + C$

Reverse.

Postfix.

postfix.

Reverse o/p.

Step 1:- $C + (B * A)$

ch.	Stack	postfix
C	—	C (operand.)
+	+	C
(+(C
B	+(CB
*	+(*	CB*
A	+(*	CBA*
)	+	

(*)

Pop. stack

CBA*+

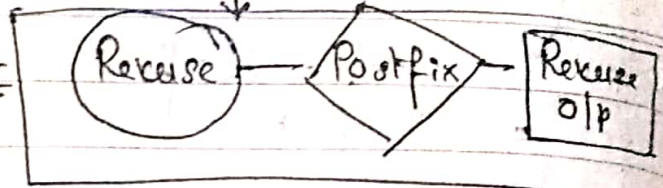
postfix.

+*ABC.

(4)

② $A + (B * C - (D / E \wedge F) * G)$

Step:-



① Reverse

$((G * (F \wedge E / D) - (*B) + A$

② Postfix

Ch	Stack	Postfix
((G
*	(*	G
(((*	G
F	((*(GF
^	((*(^	GF
E	((*(^E	GF E
/	((*(^/	GF E ^
D	((*(^/D	GF E ^ D
)	((*(^/D)	GF E ^ D
-	((*(^/D)-	GF E ^ D
(((*(^/D)-*	GF E ^ D
*	((*(^/D)-(*	E - II -
B	((*(^/D)-(*B	GF E ^ D / B
)	((*(^/D)-(*B)	GF E ^ D / B *
+	((*(^/D)-(*B)+	GF E ^ D / * CB * - A

operand


(Note -

^ > /
pop kr dege ^ > /
operon hai 1 se

$+A - * B C * / D \wedge E F G$

5
to Postfix (left to Right)

Prefix: $-, *, 9, 2, 3, |, 2, 7, 3.$
 \leftarrow right to left.

Post: $(A+B) * C$


1
27
3

$$9 \div 27 \div 3 = 9$$

+
2
3
9

[Top element in the stack]

$$2 + 3 = 5$$

-
45
9

$$9 * 5 = 45$$

*
9
5
9

$$45 - 9 = 36$$

Example of Prefix to Infix Conversion:-

$* + a - bc \mid - de + - fgh$ Infix Conversion.
 \leftarrow right to left.

-
a
b

+
f
g
h

-
d
e

d
e
(f-g)+h

/
d-e
(f-g)+h

b
c
(d-e)/(f-g)+h

b-c
(d-e)/(f-g)+h

④

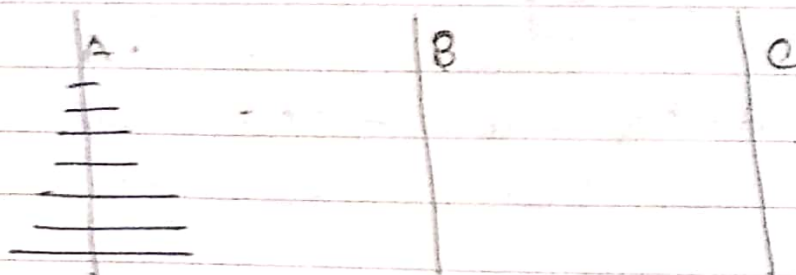
$$\begin{array}{|c|} \hline + \\ \hline a \\ \hline b - c \\ \hline (d - e) / (f - g) + h \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline * \\ \hline a + (b - c) \\ \hline (d - e) / (f - g) + h \\ \hline \end{array}$$

$$[a + (b - c) * (d - e) / (f - g) + h]$$

Tower of Hanoi

It is a mathematical puzzle.



✓ The objective of the puzzle is to move the entire stack to another rod.

(Rules) Only one disk may be moved at a time.
= Each move consist of taking the upper

2

disk from one of the Rod and sliding it onto another rod, on the top of other disk that may already be present on that Rod.

No disk may be placed on the top of smaller disk.

8

Application of Stack

Evaluation of postfix Exp.

1) Read the given exp from left to Right until the stack is empty & repeat step from 2 to 3.

② If an operand is found, push it on stack.

③ If an operator is found then.

- (1) Pop 2 top element from stack.
- (2) Evaluate the expression formed by
- (3) 2 operands & operator.
- (4) Push the Result back to back.

Set Result equal to top element of the stack

Exit

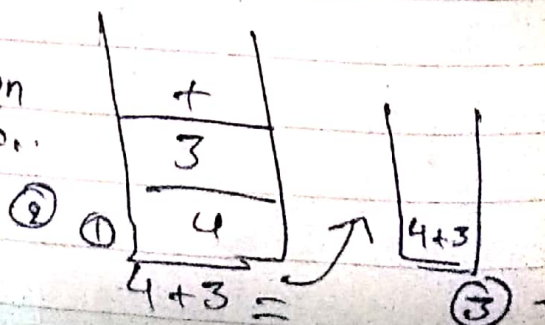
Regular grammar to
CFA.

fSA to fsm.

Non-deterministic
TM + encounter on
deterministic operators.

operand → push it on stack

3 con →

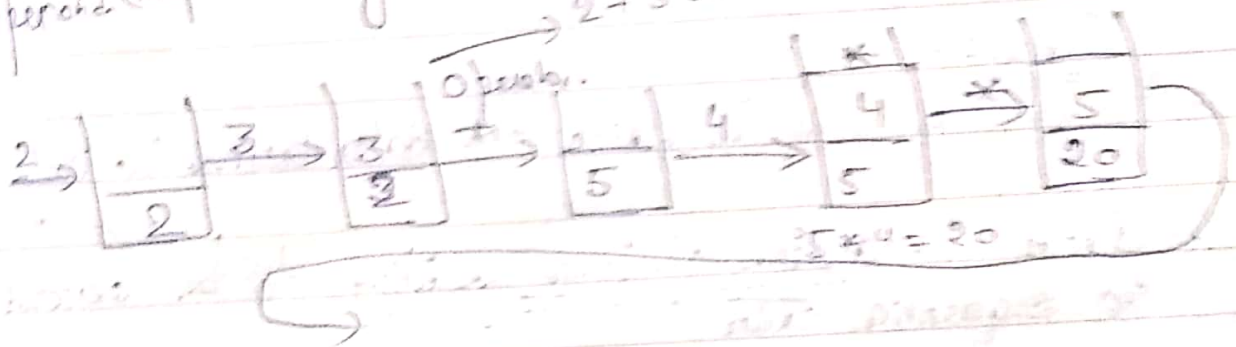


9

Evaluation of postfix Expression

① $AB + C * D$ where
 $A = 2, B = 3, C = 4$ & $D = 5$.

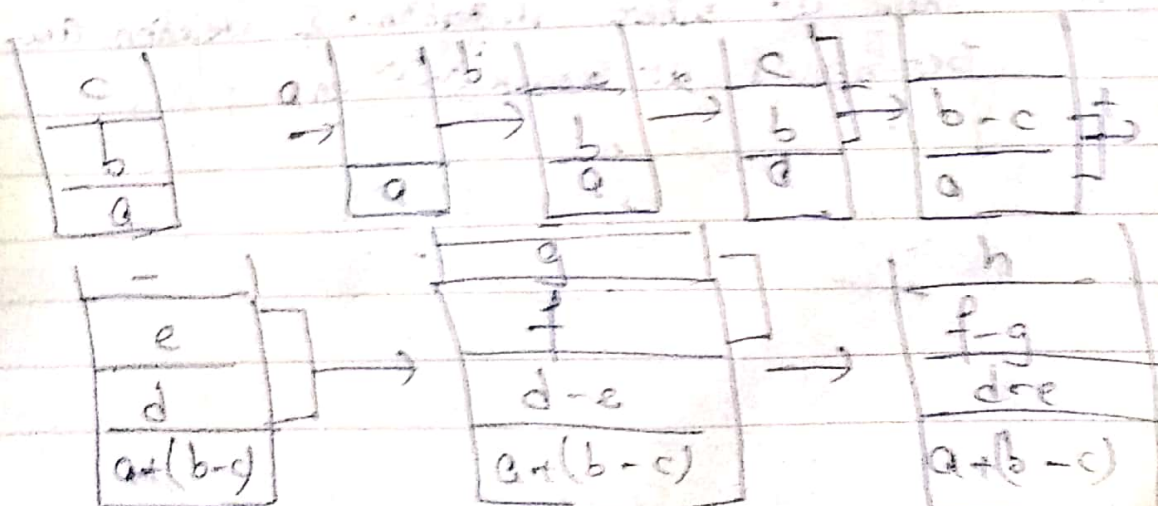
$2, 3 + 4 * 5$
 Operands (left to right)

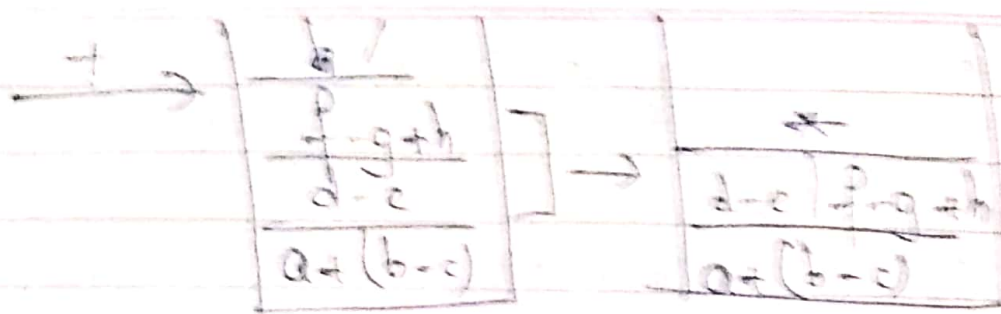


$\frac{1}{4} = 0.25$

② Eg of postfix to Infix conversion

$abc + de - fg - h + / *$





$$a+(b-c) \mid * \mid d-c \mid f-g+h$$

Queue:-

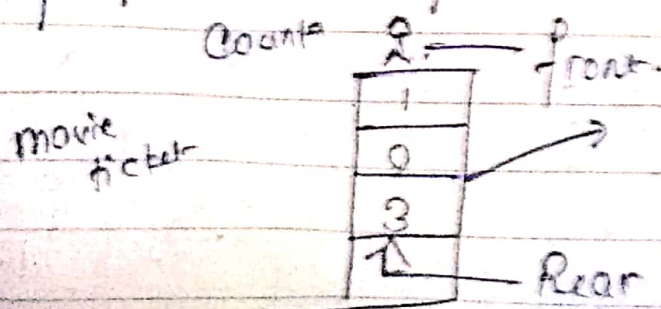
1) Queue, the order in which the data arrives is imp.

2) Queue is a line of items waiting to be served in sequential order.

* A queue is an ordered list in which insertion are done at one end (Rear) & deletion are done at other end (front).

* Working principle is fifo.

* Linear ds where insertion & deletion are performed at separate Rear end & front end.



Queue full

(11)

$R = R + 1$

$f = f + 1$ no element

Insertion of Element

- 1.) Initialize $f = 0, R = -1$
- 2.) Check overflow condition
If $f = 0$ & $R = \text{MAXSIZE}$ or
 $f = R + 1$
then write overflow & exit.
- 3.) If $F = \text{Null}$;
Set $f = 0$ & $R = 0$
else if $R = \text{MAXSIZE}$
Set $R = 0$
- 4.) Set $R = R + 1$) Increment
Real counter.
- 5.) $\text{Queue}[R] = \text{Item}$
Exit

Deletion of Element

- i.) Check Underflow condition
if $f < 0$, write underflow
exit.
- ii.) Set item = Queue[f]
- iii.) if $F = R$
then Set $F = R = \text{Null}$
else if
 $F = \text{MAXSIZE}$
then Set $F = 0$
// (Setting the variable)
- iv.) Set $F = F + 1$
- v.) Exit.

4 | 5 | 6 | 7 | 8 | 9

$f = 0$ $R = 0$ (as variable).

Initial level = 0.

front =

$f = 0, R = 0$

4 Remove.

$f = \text{increment}$

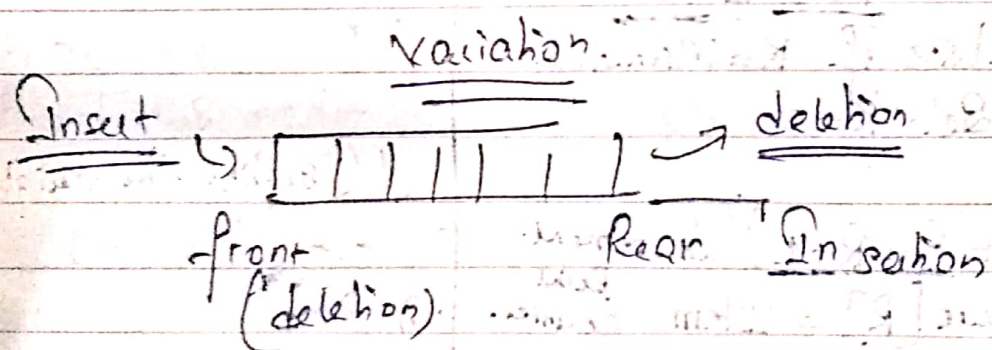
$f = 1$
item $Q(1) = 4$.

Deque (Double Ended Queue)

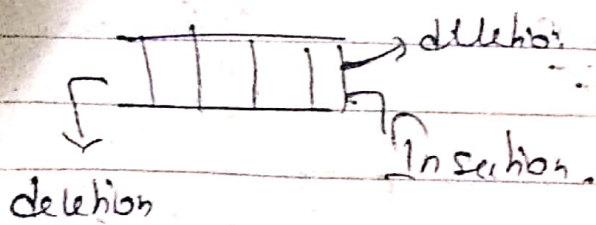
In this insertion & deletion can be performed at both end

ie. front pointer can be used for insertion &

Rear pointer can be used for deletion.



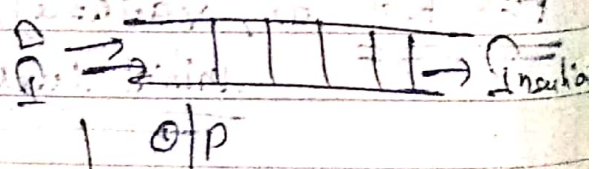
Input Restricted Queue



Deletion can be performed at both end

Insertion at one end (Rear)

Output Restricted Queue



deletion at one end (front)

Insertion can be performed at both ends.

(13)

Queue (FIFO) and priority queue

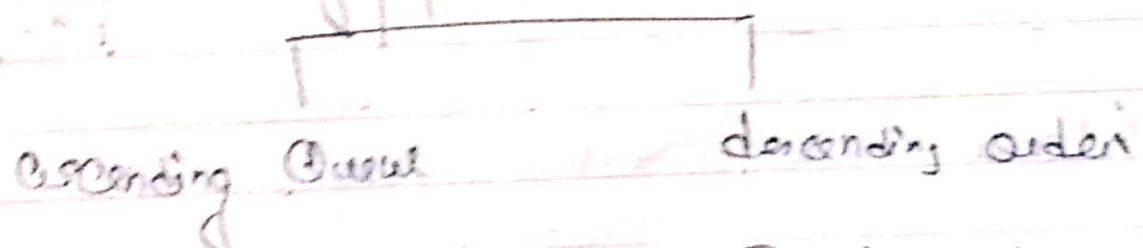
PRIORITY QUEUE :- (VIP - movie) enter 1st in queue.

Each element is inserted / deleted on the basis of their priority.

→ Highest priority > Lower priority.

→ Same priority [FCFS basis].

Type of PQ.



[Lower priority no. to high no.]

[Higher priority no. to high no.]

- ①
- ②
- ③
- ④
- ⑤

- ⑤
- ④
- ③
- ②
- ①

Rep of priority queue

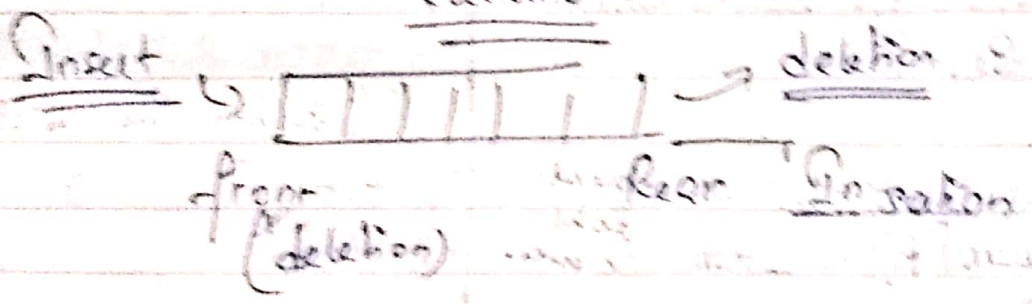
Queue (Double Ended Queue)

In this insertion & deletion can be performed at both end

ie. front pointer can be used for insertion &

rear pointer can be used for deletion.

Variation



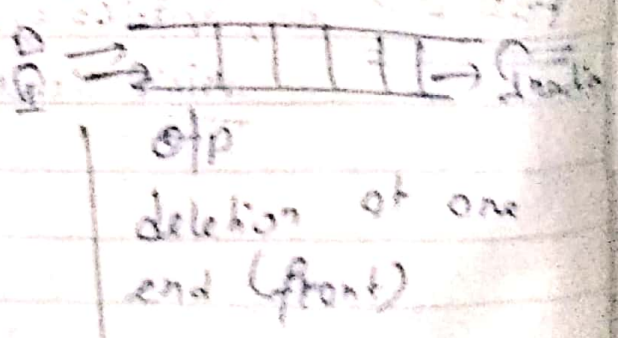
Input Restricted Queue

O/P Restricted Queue



Deletion can be performed at both end

Insertion at one end (rear)



Insertion can be performed at both ends.

9th - Insertion - 1

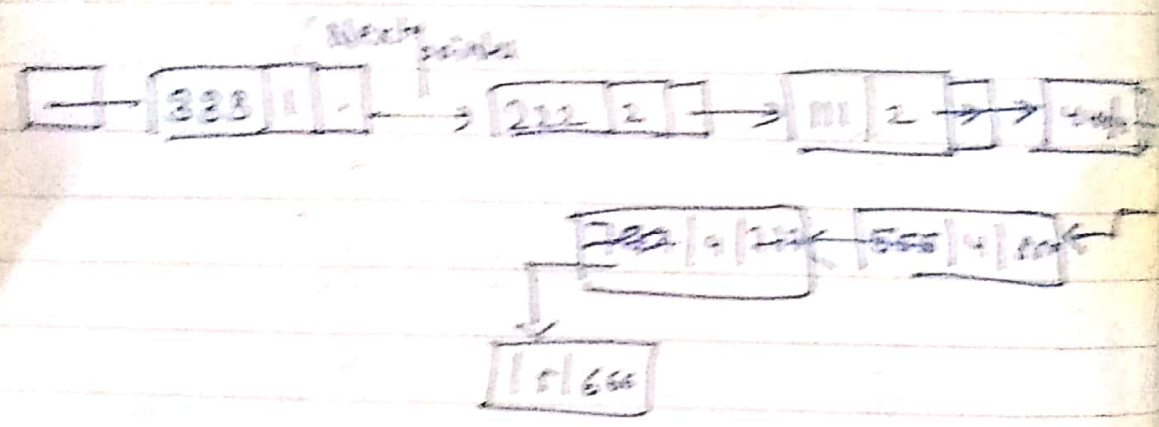
(14)

Priority

Priority no.

Ques	Ques	Link	Link
1 321	1	② = 2nd (Pcp)	6
3 456	0	2nd & 1st	4
4 755	0		9
5 322	1		1
6 111	2		3
7			
8 666	8		
9 222	0		8

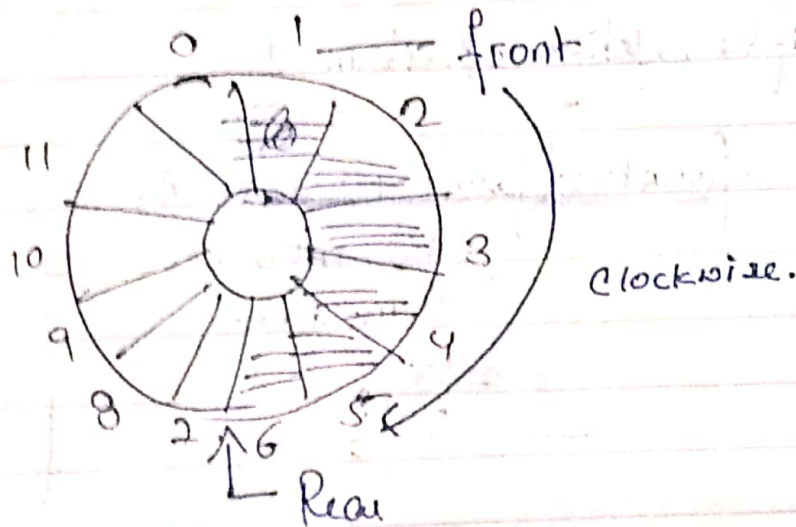
Low priority no = High priority



Rear = Rear + 1

Circular queue.

In this the Rear pointer can point at the beginning of the queue when it reaches end of the array.



Adv:- empty space can be fill again using Rear pointer.

Initialization:-

$$\begin{aligned} \text{front} &= 0; \\ \text{Rear} &= 0; \end{aligned}$$

Insertion

$$\text{arr}[\text{Rear}] = \text{item};$$

$$\text{Rear} = (\text{Rear} + 1) \bmod n;$$

(total no. of space in circular queue)

Deletion

$$\text{item} = \text{arr}[\text{front}];$$

$$\text{front} = (\text{front} + 1) \bmod n;$$

16 $n = 12$

after 6.
 $(6+1) \bmod n$
 $7 \bmod 12$

$n = 12$

$(6+1) \bmod n$
 $7 \bmod 12$
 $= 7$

Increment normally.

after deletion of item 2.

$front = (front + 1) \bmod n$
 $(2+1) \bmod 12$
 $3 \bmod 12$
 $= \underline{\underline{3}}$

Rear = 11

diff when reaches at the end of queue.

11 

$(11+1) \bmod 12$
 $12 \bmod 12$

$= 0$

Rear is pointing to the first location after reaching end.