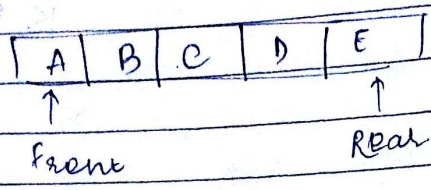
16/08/19

QUEUE

① LINEAR

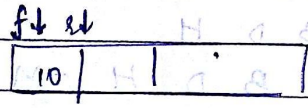


Insertion occurs at Rear end.
Deletion occurs at Front end

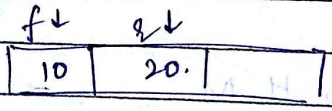
Rear and Front at '-1' ⇒ Empty Queue.
Maxsize → Maximum size of queue.

→ Insertion into linear queue:-

Ist element :-
item = 10;
rear = rear + 1
Queue[rear] = item;
Set front = 0

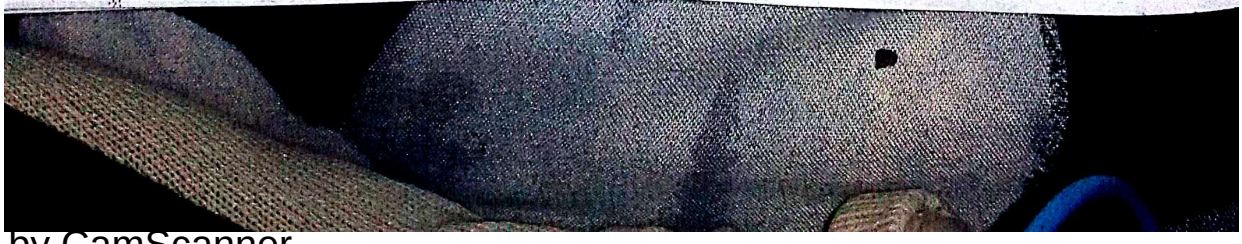


2nd element :-
rear = rear + 1
Queue[1] = 20
Front doesn't change.



Condition for OVERFLOW of queue

```
If (rear >= MAXSIZE - 1)
{
    Print "Overflow";
}
```



MBD WRITEWELL
Date .....
Page .....

PROGRAM Program for insertion into linear queue:-

Algorithm :-

Qinsert (Maxsize, item)

Step 1: Initialize front = -1
rear = -1

Step 2: [Check for Overflow]
If Rear >= Maxsize - 1
Print ("Overflow")
return

Step 3: else
(Increment Rear)
Rear = Rear + 1

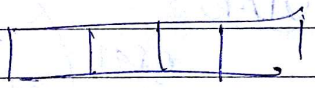
Step 4: [Insert element]
Queue [Rear] = item

Step 5: [check for front]
If front = -1
set (front = 0.)

Step :- Repeat step ③ and ④ until ② false becomes false

Step 6: Exit

DRY RUN MAXSIZE = 4



① f = -1, r = -1

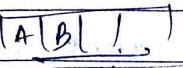
-1 >= 4-1 => -1 >= 3 FALSE

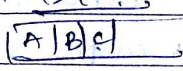
Skip to else ③.

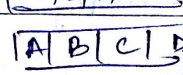
f=0, r=0

Q[0]

Repeat step ⑤ and ④

0 >= 3 

1 >= 3 

2 >= 3 

3 >= 3 → TRUE

Print "Overflow"

Rear - Front + 1 = No. of elements present at any instant

MBD WRITEWELL
Date .....
Page .....

① PROGRAM:

For deletion of elements from linear queue.

Q- delete (dequeue) the

Step 1: If front = = -1
Print "Underflow"
Return

Step 2: item = Queue [front].

Step 3: (Print the value to be deleted)
Print "item".

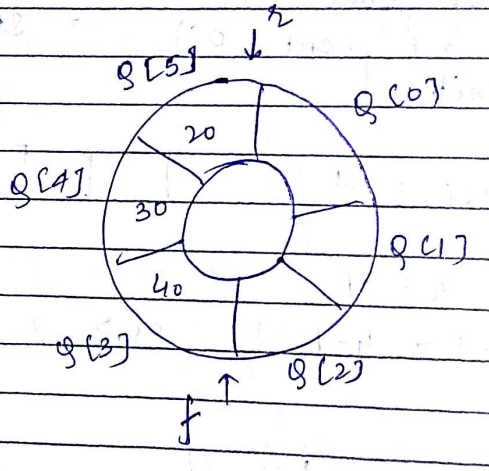
Step 4: (Increament front)
front = front + 1

Step 5: Repeat until
front = = rear.

Step 6: Else set
front = -1 , rear = -1

Step 7: Exit.

② CIRCULAR



Drawback of linear queues -

10, 20, 30, 40, 50, 60

| 10 | 20 | 30 | 40 |

→

| | | 30 | 40 |

Elements cannot be further added to the empty columns.

MBD WRITEWELL
Date .....
Page .....

PROGRAM :- For insertion of elements in circular queue

Q insert (MAXSIZE, item)

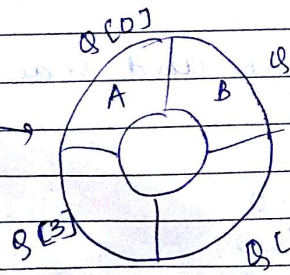
Algorithm :-

① If (front == (Rear+1) % MAXSIZE)
Print "Overflow"
Exit

② else take value
If (front == -1)
Set front = Rear = 0.
else
set Rear = ((Rear+1) % MAXSIZE).

③ [Assign value]
Queue[Rear] = item

④ Exit



No. of elements = front + Rear + 1

PROGRAM :- Q Delete (MAXSIZE, item)

Algorithm :-

① If (front == -1)
print "Underflow"
return.

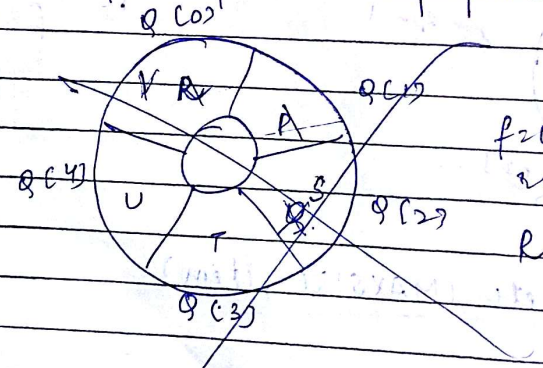
② else item = queue[front]
if (front == Rear)
set front = -1
rear = -1
else
front = (front + 1) % MAXSIZE

NUMERICAL

Consider the foll circular queue of characters which is allocated 5 memory slots. Describe the foll operations :-

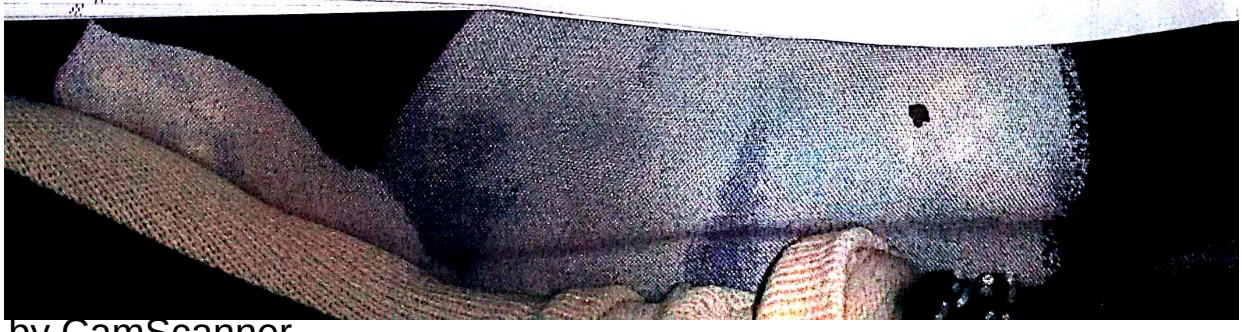
a) R is added to queue
b) P and Q are added to the queue
c) Q is deleted
d) S, T, U and R are added to the queue
e) 2 letters deleted
f) 'V' is added to the queue.

Find the final positions of front and rear.



r = 0
f = 2

Initial, r = -1
f = -1



② else item = queue[front]
if [front == Rear
set front = -1
rear = -1

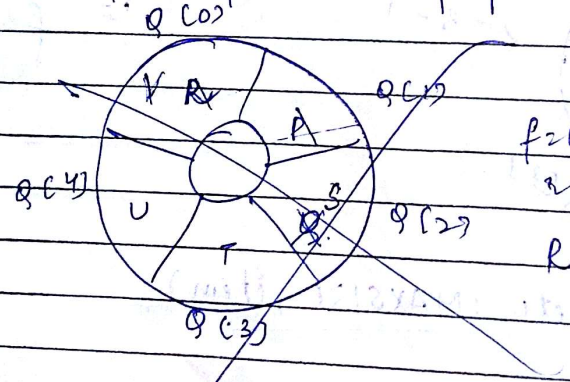
else
front = (front+1) % MAXSIZE

NUMERICAL

Consider the foll circular queue of characters which is allocated 5 memory sites. Describe the foll operations :-

a) R is added to queue
b) P and Q are added to the queue
c) 2 letters deleted
d) S, T, U and R are added to the queue
e) 2 letters deleted
f) 'V' is added to the queue.

Find the final positions of front and rear.

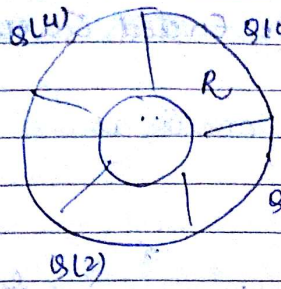


r = 0
f = 2.

Initial, r = -1
f = -1

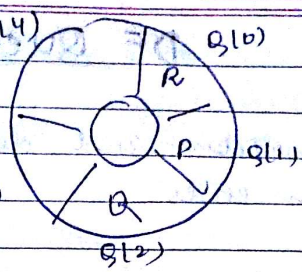
MBD WRITEWELL
Date .....
Page .....

Step a)



f = 0
r = 0

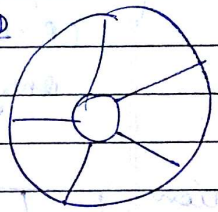
Step b)



Rear = (Rear + 1) % MAXSIZE
0 = (0 + 1) % 5
= 1
1 = (1 + 1) % 5
= 2 % 5
Rear = 2. f = 0.

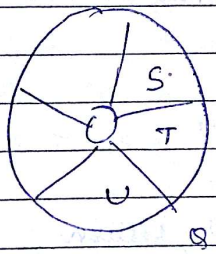
Step c) Deletion takes from FRONT END

front = (f + 1) % MAXSIZE
= 1 % 5.
= 1

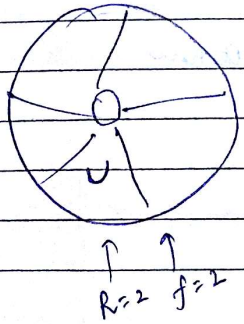


R is deleted f = 1
P is deleted f = 2
Q is deleted f = -1, r = -1 (empty queue)

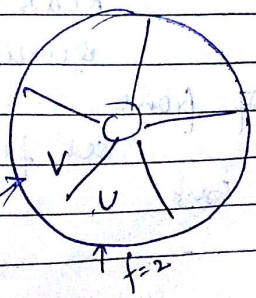
Step d)



Step e) 2 times deleted

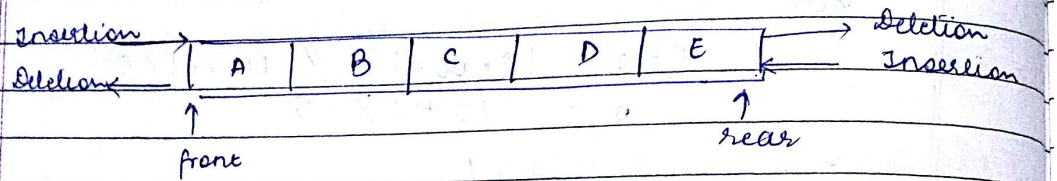


f) V is added



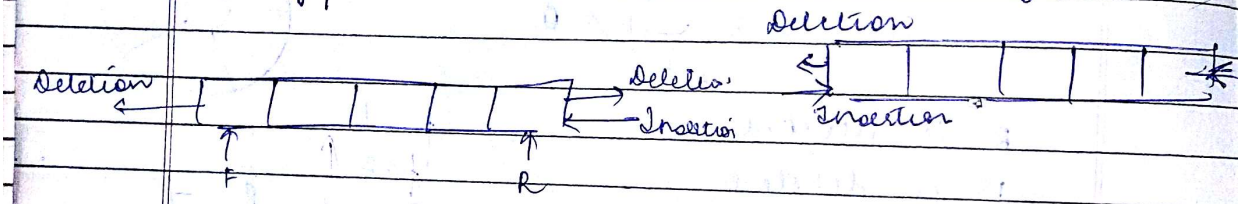
(3) DE Queue (Double Ended Queue)

→ Insertion and deletion can occur at both ends



(a) Input - restricted (R-end will be used only for insertion)

(b) Output - restricted (F-end will be used only for deletion)



PROGRAM → Insertion of element from DE-Queue at Rear End

Algorithm

1. Set front = -1 and Rear = -1
2. If Rear = Maxsize - 1
   Print ("Overflow"); return;
3. Else
   Rear = Rear + 1
   Queue [Rear] = item
4. If front = = -1
   Set front = 0
5. Exit

MBD WRITEWELL
Date .....
Page .....

Deletion of element from Deletion (Output queue restricted)

Algorithm

1. If front == -1
   Print ("Underflow")
2. Else item = Queue [front]
   print ("item)
3. front = front + 1

Insertion from front end

A	B	C	

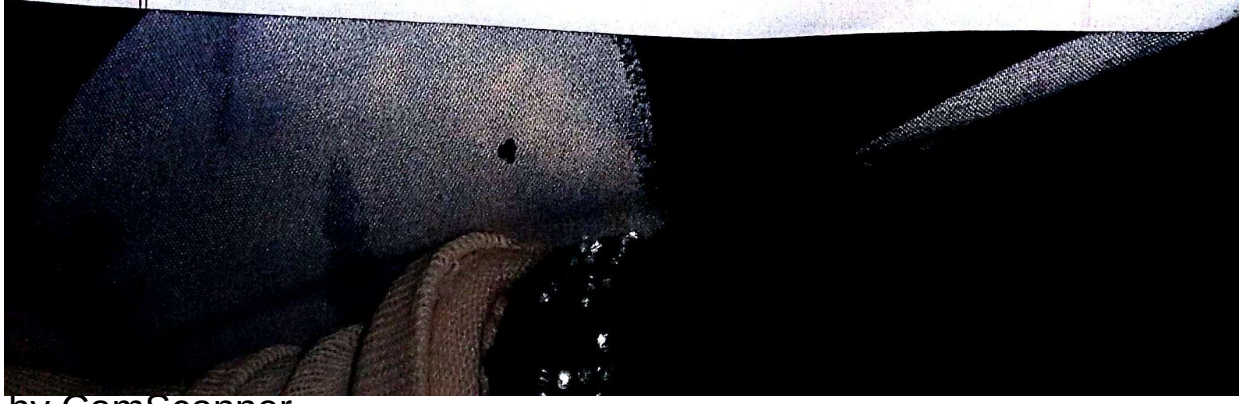
f = 0, r = 2

→ Insertion

If D is to inserted,

D	A	B	C

1. (check for space at the front end)
   If ((~~f == 0 && r == 0~~)
   ~~If (Rear == 0) &&~~ (
2. If (Front == 0) && (Rear == MAXSIZE - 1)
   || (front = Rear + 1))
3. : Print "overflow"
4. else (insert item at front)
   front = front - 1
   Q[Queue [front] = "item



Date .....
Page .....

Program: Deletion from Rear End

① If [Check for ("Underflow")]
if rear = = -1
Print "Queue is Empty"

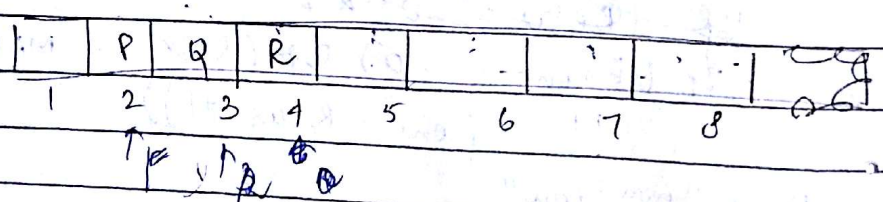
② Else
item = Q[Rear]
Rear = Rear - 1

③ Exit

For printing element;

i = front to i = rear - 1
i++;
Total no of elements;

Qu) Perform the foll. operation on a circular de-queue.



(i) deletion of last element
(ii) insertion of element 'O' at front end
(iii) insertion of elements R, S, T at the rear end
(iv) insertion of element U and V at front end
(v) insertion of element 'W' at rear end
(vi) Insertion of element 'X' at the left end.

MBD WRITEWELL
Date ....
Page ....

1. Deletion of last element
2. O → f
3. R, S, T → R
4. U, V → f
5. I → R
6.
7. N → f

1. Rear = (Rear - 1) TRUE
= 3 - 1 = 2 / 8 TRUE
= 0.

| P | Q | R | S | |

F R

| . | O P | Q | R | S | T | U | V |

Further insertion not required.

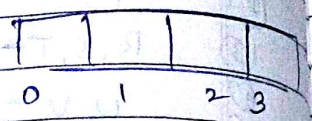
There is an input restricted PQ, Deque & elements are:
PQ 1. ~~Student~~
2. 6, 4, 5
3. 4, 3
4. Delete from front
5. Delete from rear.

(~~Output~~ restricted) I/P

6	4	5	.
0	1	2	3

Scanned by CamScanner

There is an empty output restricted queue.
Insert 10, 20, 30.
Insert 40 at front.



19/08/16

PRIORITY QUEUE

Operation	Linear Queue	circular Queue
Overflow	Rear >= Maxsize - 1	front == (Rear+1) % Maxsize OR (front == 0 && Rear == Maxsize-1) || front == Rear+1
Underflow	front == -1	front == -1
Insertion	rear == rear+1 {	rear = (rear+1) % Maxsize OR if (front == maxsize - 1) rear = 0 else rear = rear+1
Deletion	front = front+1	front = (front+1) % maxsize

Priority Queue insertion in circular queue-

1. [Check if queue is full.]
   If front == (Rear+1) % Maxsize
   Print("Overflow")
   Exit
2. else (front == -1)
   set
   front = rear = 0

else
rear = (rear + 1) % Maxsize

Circular Priority queue (Insert)

→ ① [check if queue is full]
If ((front == 0 && Rear == max size - 1) || (front == Rear + 1))
print ("Queue is full")

② [check whether Rear points to the last position of Queue]
If (Rear == max size - 1) then
Rear = 0
else if (Rear == -1)
Rear = front = 0
else
Rear = Rear + 1

rear = (rear + 1) % Maxsize

③ [Insert new element]
P. queue [Rear] = Item

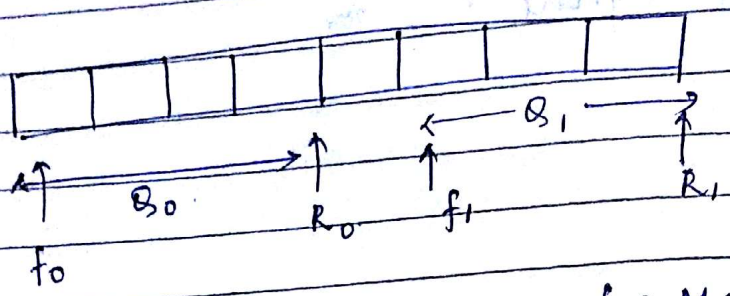
④ [sort the element of queue in Ascending or descending order]
sort (f sort, Rear, PQ)

⑤ Exit

23/08/16

Date .....
Page .....

MULTIPLE QUEUE



$f_0 = -1$
$R_0 = -1$

$f_1 = \text{Maxsize}$
$R_1 = \text{Maxsize}$

Insertion into a Multiple Queue :-

(1) [check for overflow]
If (count == MAXSIZE)
print ("Overflow")
return

(2) else
[check in which queue item is to be inserted]
If (queue no. == 0)
then if ($q.rear_0$ == -1)
$q.rear_0 = q.front_0 = 0$

(3) else
$q.rear_0 = q.rear_0 + 1$
$q.num[rear_0] = item$

(4) else
$q.rear_1 = q.rear_1 - 1$
$q.num[rear_1] = item$

(5) Exit

Algorithm for multiple queue deletion :-

① [check for underflow]

If (count = = 0)
print ("Underflow")
return

If (front$_0$ = = -1 && front$_1$ = = maxsize)

② else

[check from which side item is to be deleted]

If (queue no. = = 0)
then (check if front$_0$ = = -1)
item = q [front$_0$].
front$_0$ ++

front 1 = front 1 - 1

If front = = rear
last element is deleted
If front$_1$ = rear$_1$ = Maxsize
queue 2 is empty.

③ else

[check from which side item is to be deleted]

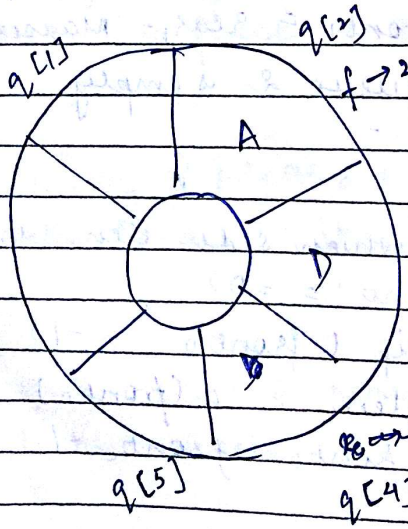
If (queue no. = = 0)
check if (front$_0$ = = -1)
item = q [front$_0$]
front$_0$ = front$_0$ + 1

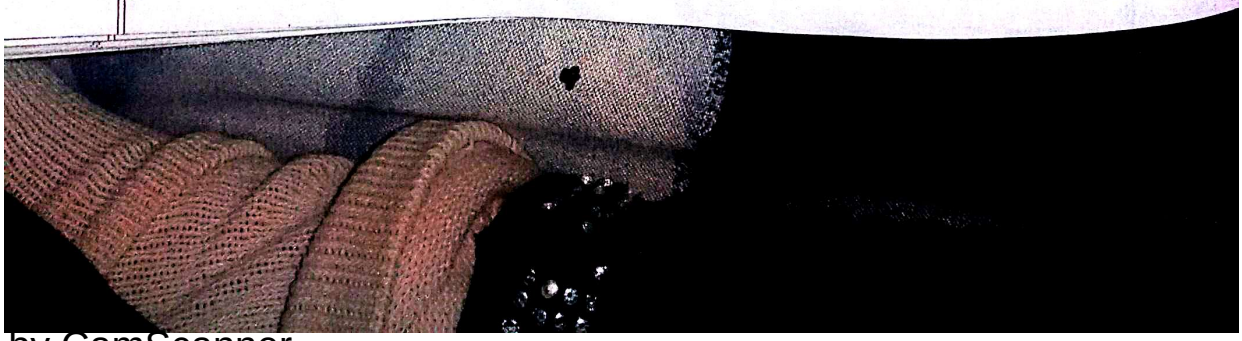
Applications of Queue :-

1. Client - server routines
   (order of requesting = order of processing)
   Eg :- Printing of documents.
2. Multi-processing, batch processing,
3. CPU scheduling
4. Round-Robin scheduling
5. Daily routine :- Movie ticket, reservation, railway/metro reservation

A circular queue of _, A, D, —, —,
front = 2, rear = 3
Operations to be performed :-

a) Add S
b) Add J
c) Delete two letters
d) Shift towards left to bring all free spaces to right
e) Insert M, H, I and delete one letter.



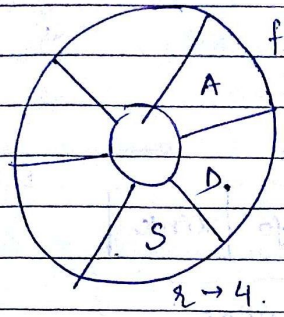


MBD WRITEWELL

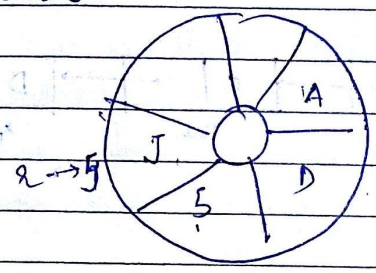
Date .....

Page .....

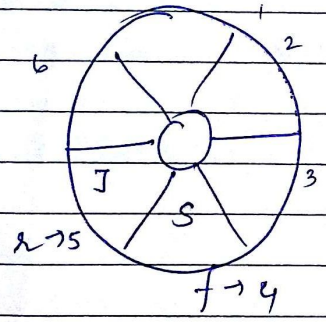
a) Add S



(b) Add J



(c) Delete 2 letters



_ _ _ S J _

(d) S J _ _ _ _