14, 27, 33, 35, 10.

(c) Compare 33 and 35, since 33 < 35, no swapping.

(d) Compare 35 and 10, since 35 > 10, swapping required.

14, 27, 33, 10, 35.

(Largest element placed at the highest index of array)

Pass-2:-

(a) Compare 14 and 27, since 14 < 27, no swapping.

(b) Compare 27 and 33, since 27 < 33, no swapping.

(c) Compare 33 and 10, since 33 > 10, swapping required.

14, 27, 10, 33, 35

(Second largest element placed at 2nd highest position).

Pass-3:-

(a) Compare 14 and 27, since 14 < 27, no swapping.

(b) Compare 27 and 10, since 27 > 10, swapping required.

14, 10, 27, 33, 35

↳ Third highest element placed.

Pass-4:-

(a) Compare 14 and 10, since 14 > 10, swapping required.

10, 14, 27, 33, 35.

↳ Fourth highest element placed.

10, 14, 27, 33, 35.

Sorted list.

Unit - 2.

1) Consider a circular queue of size 5. Perform the following operations in the specified order and write front & rear values after every operation and also display the elem contents.

i) insert 10, 20, 30 and ii) delete 2 elements.

iii) insert 50, 60, 70 and 80 iv) delete 2 elements.

Ans:- Circular queue:-

Since, size = 5, So a circular queue of 5 elements.

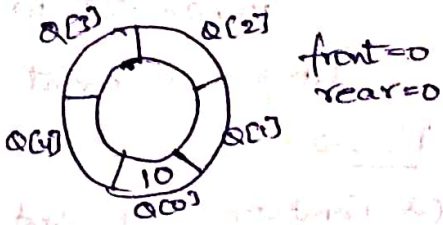
Initially, front = rear = -1

(i) insert 10, 20, 30

Step-1:- if Front = 0 and rear = max-1, its overflow.
here front ≠ 0 and rear ≠ max-1, so proceed.

Step-2:- if front = -1 and rear = -1
So, front = rear = 0.

Step-3:- and now Q[rear] = val.
Q[0] = 10.

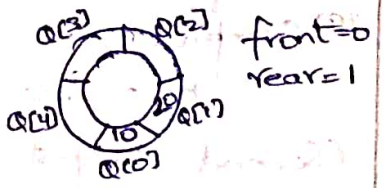


for ~~20~~ inserting 20,

Step-1:- Front ≠ 0 and rear ≠ max-1, so proceed.

Step-2:- here rear ≠ max-1.
So, set rear = rear+1 = 0+1 = 1

Step-3:- & Q[rear] = val, Q[1] = 20.

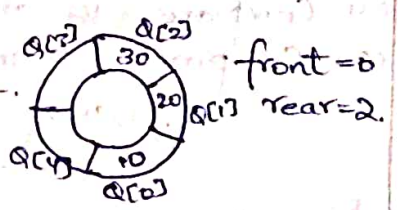


for inserting 30,

Step-1:- front = 0 and rear ≠ max-1, so proceed.

Step-2:- here rear ≠ max-1.
So, set rear = rear+1 = 1+1 = 2.

Step-3:- Q[rear] = val, Q[2] = 30



(ii) delete two elements

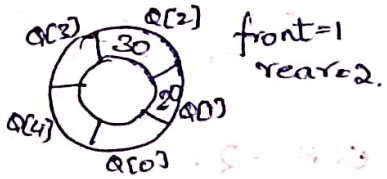
Deletion is done at front end. (first inserted is first deleted)

delete element 10 :-

Step-1:- front ≠ -1, so not underflow, proceed.

Step-2:- set val = Q[0] = 10.

Step-3:- here front ≠ rear & front ≠ max-1
So front++ ⇒ front = 0+1 = 1
element 10 deleted.

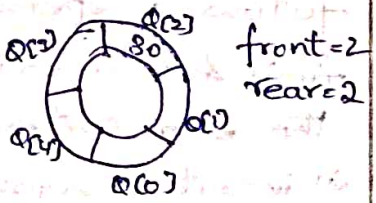


delete element 20:-

Step-1:- front ≠ -1, so not underflow, proceed.

Step-2:- set val = Q[1] = 20.

Step-3:- here front ≠ rear & front ≠ max-1.
So front++ ⇒ front = 1+1 = 2.
element 20 deleted.



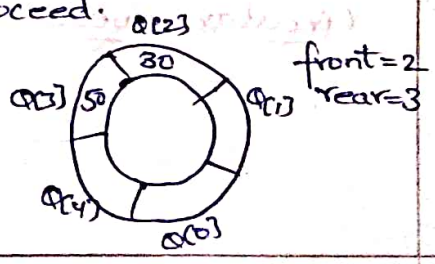
(iii) insert 50, 60, 70, 80

Inserting 50.

Step-1:- front ≠ 0, & rear ≠ max-1, so proceed.

Step-2:- here rear ≠ max-1, so
rear++ ⇒ rear = 2+1 = 3.

Step-3:- Q[rear] = val, Q[3] = 50



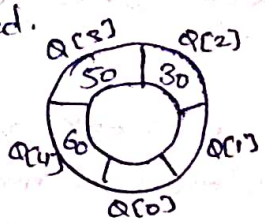
inserting 60.

Step-1:- front ≠ 0 & rear ≠ max-1, So proceed.

Step-2:- here rear ≠ max-1, So

rear++ => rear = 4.

Step-3:- Q[rear] = val, Q[4] = 60



front = 2
rear = 4

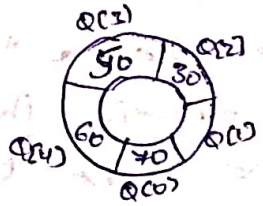
inserting 70.

Step-1:- front ≠ 0, so proceed even rear = max-1

Step-2:- here rear = max-1 & front ≠ 0,

so set rear = 0.

Step3:- Q[rear] = val, Q[0] = 70.



front = 2
rear = 0

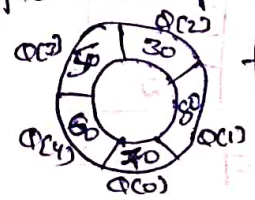
inserting 80:

Step-1:- front ≠ 0 & rear ≠ max-1, & rear+1 ≠ front, so proceed

Step 2:- here rear ≠ max-1 so,

rear ++ => rear = 1

Step-3:- Q[rear] = val, Q[1] = 80



front = 2
rear = 1

(iv) delete two elements

Deletion is done at front end.

delete element 30:-

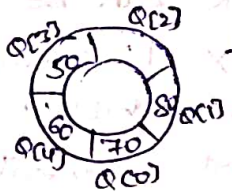
Step-1:- front ≠ -1, so proceed

Step-2:- set val = Q[front] = Q[2] = 30

Step-3:- here front ≠ rear & front ≠ max-1.

so front ++ -> front = 3

element 30 deleted.



front = 3
rear = 1.

delete element 50:-

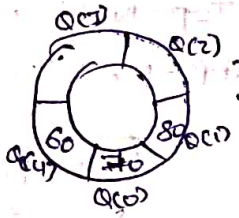
Step-1:- front ≠ -1, so proceed.

Step-2:- set val = Q[front] = Q[3] = 50.

Step-3:- here front ≠ rear & front ≠ max-1

So front ++ => front = 4.

element 50 deleted.

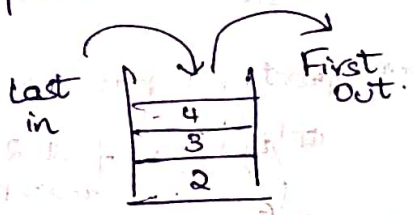


front = 4
rear = 1.

(2) Write about stack and its applications.

Ans:- Stack:-

Stack is a linear data structure which stores elements in an ordered manner like a pile of plates. The elements in a stack are added and removed only from one end which is called the "Top". Stack is called a "LIFO" (Last in First Out) data structure, as the element that was inserted ~~first~~ last is the first one to be taken out.



Uses of stack:- In order to keep track of returning point of each active function.

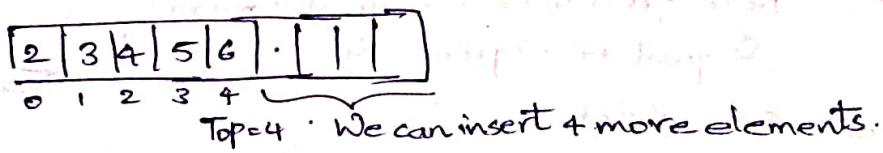
Array representation of Stack:-

→ Stack can be represented as a linear array.

→ Every stack has a variable called 'Top' associated with it, which is used to store the address of the topmost element of the stack.

→ There is another variable 'MAX' to store maximum no. of elements that the stack can hold.

If Top=NULL, it indicates that the stack is empty and if Top=MAX-1, means stack is full.



Operations of a stack:-

push():- push operation adds an element to the top of the stack.

Pop():- pop operation removes an element from the top of the stack.

Peek():- peek operation returns the value of topmost element of the stack.

Applications of a stack:-

(1) Reversing a list.

stack is used to reverse a string. We push the characters of string one by one into stack and then pop character from stack.

(2) Paranthesis Checker:- Stack is used to check the proper opening and closing of paranthesis.

(3) Evaluation of a prefix expression.

(4) Evaluation of a postfix expression.

An expression can be represented in prefix, postfix or infix notation. Stack is used to evaluate prefix, postfix and infix expressions.

infix:- A+B , prefix:- +AB , postfix:- AB+.

(5) Conversion of an infix expression into a postfix expression.

(6) Conversion of an infix expression into a prefix expression.

Stack can be used to convert one form of expression to another.

(7) Recursion.

When we call a function call ~~and return~~ from one other function, that function call statement may not be the first statement. After calling the function, we also have to come back from the function area to the back from the function area to the place where we have left our control. So, we store the address of PC into the stack, then goto function body to execute it. After completion of execution, it pops out add. from stack & assign it into PC to resume the task again.

(3) What is Queue? Explain about circular queue with examples.

Ans:- Queue:- A queue is a FIFO (First in First out) data structure in which that element that is inserted first is first one to be taken out.

The elements in queue are added at one end called the "REAR" and removed from the other end called the "FRONT".

Queues can be implemented using arrays / linked lists.

Array Representation of Queues:-

Queues can be easily represented using linear arrays. Every queue has FRONT and REAR that point to the position from where deletions and insertions can be done.

12	9	7	18	14	36				
0	1	2	3	4	5	6	7	8	9

FRONT=0 REAR=5

Insertion is done at Rear end & deletion at front end.

~~In the~~ When REAR= MAX-1, then overflow condition.

When FRONT & REAR = -1, then underflow condition.

In the above example, FRONT=0, REAR=5, and suppose we want to add one element, we must increment "REAR" and then value would be stored at the position pointed by REAR.

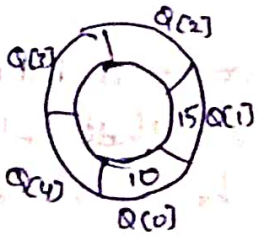
Circular Queue:- In circular queue, the first index comes right after the last index.

Circular queue.

For insertion, we have to check the following conditions front=0 and rear=max-1 (or) rear=front-1, then the circular queue is full.

Ex:- (1) Inserting an element into the circular queue.

Let the circular queue be



front=0
rear=1

Now inserting element 20 into the circular queue, insertion is done at rear end.

Algorithm:-

Step-1:- If front=0 and rear=max-1
Write "overflow".
Goto step-4.

Step2:- If front =-1 and rear=-1.
set front=rear=0.
Else if rear=max-1 and front!=0
set rear=0
else
set rear=rear+1.

Step 3:- Set Queue[rear]=val.

Step 4:- Exit.

Here,

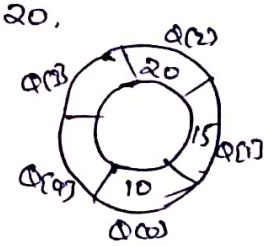
Step-1:- front=0 but rear≠max-1, so proceed

Step-2:- here rear≠max-1
So, set rear=rear+1 = 1+1 = 2.

Step-3:- set Q[rear]=val ; Q[2]=20.

Step-4:- Exit.

So, element inserted at Q[2]



front=0
rear=2.

(ii) Deleting an element from the circular queue.

~~Algorithm~~:- In the above circular queue, let the element 10 is deleted.

Deletion is done at front end.

Algorithm:-

Step-1:- If front =-1
write "underflow".
Go to step-4.

Step-2:- set Val= Queue[front]

Step-3:- if front = rear.
set front= rear= -1.
else
if front==max-1
set front=0.
else
front++.

Step-4:- Exit.

Here.

Step-1:- front ≠ -1 , so proceed , no underflow.

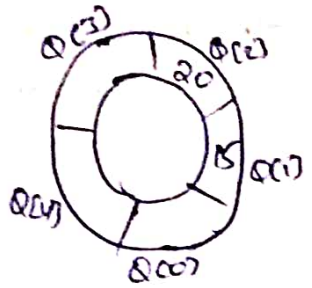
Step-2:- set val = Q[front] = Q[0] = 10.

Step-3:- here front ≠ rear and front ≠ max-1

so front ++ ⇒ front = 0+1 = 1

element 10 is deleted.

Step-4:- Exit



front = 1

rear = 2.