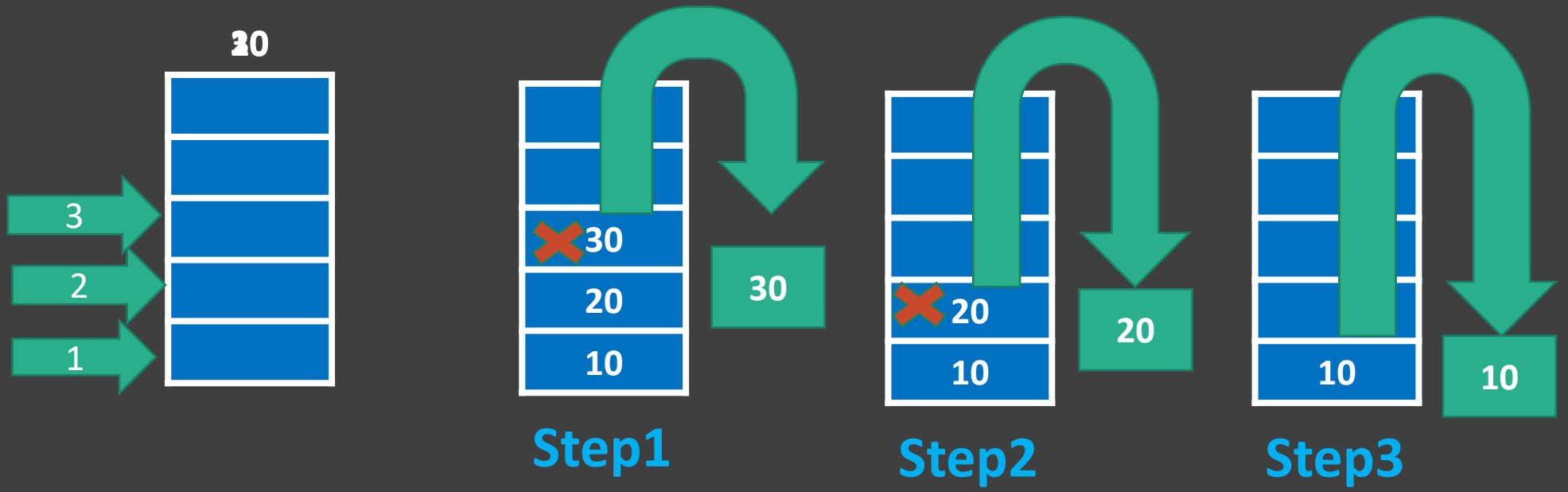
Stack Data Structure- Stack ADT

A Linear Data Structure that stores data in a **continues / sequential** passion and follows LIFO/ FILO strategy.

LIFO - Last In First Out

FILO - First In Last Out

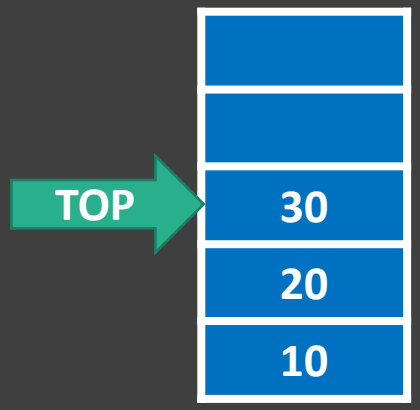


Terminology - Stack ADT

Top: It is a pointer connected to the top most index of stack

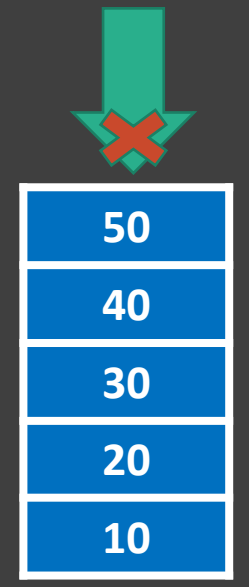
Any operation on stack can be performed only using TOP variable

Initial value of top is -1



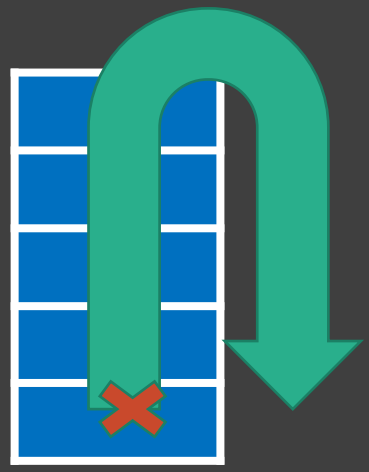
Overflow: It is a state of stack that represents stack is FULL.

We cant insert any values in Overflow state of stack



Underflow: It is a state of stack that represents stack is EMPTY.

We cant Delete any values in Underflow state of stack



Operations on - Stack ADT

1. PUSH(x)
2. POP( )
3. DISPLAY( )
4. PEEK( )
5. COUNT( )

PUSH(x) Operation:

Using PUSH operation we can insert a new value into the stack

STEPS:

Step1: Check Overflow state of the stack

if(top>=Max-1)

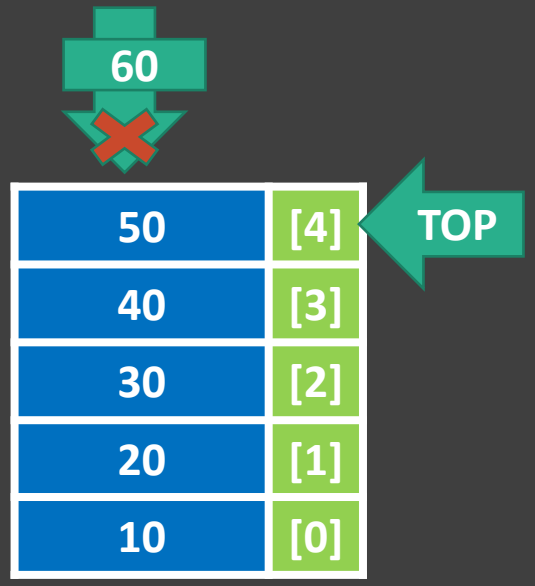
then

Overflow

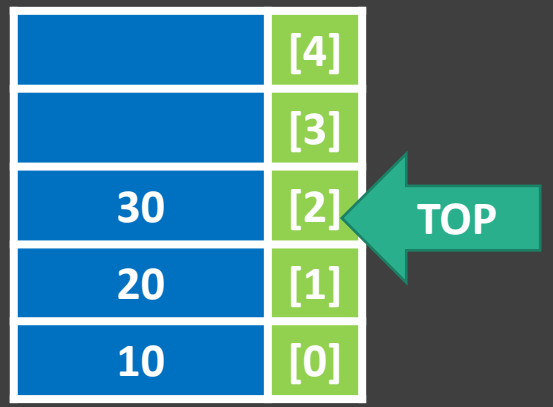
Step2: top++

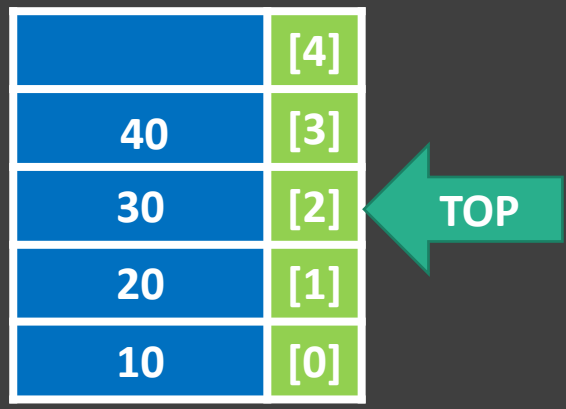
Stack[top] = x

Print x is inserted



New Value





POP( ) Operation:

Using POP operation we can delete a existing value from the stack

STEPS:

Step1: Check Underflow state of the stack

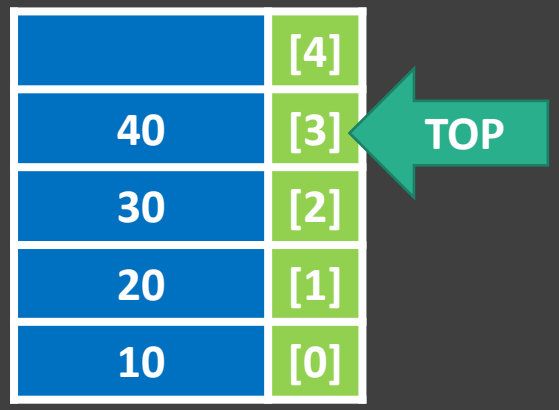
if(top==-1)

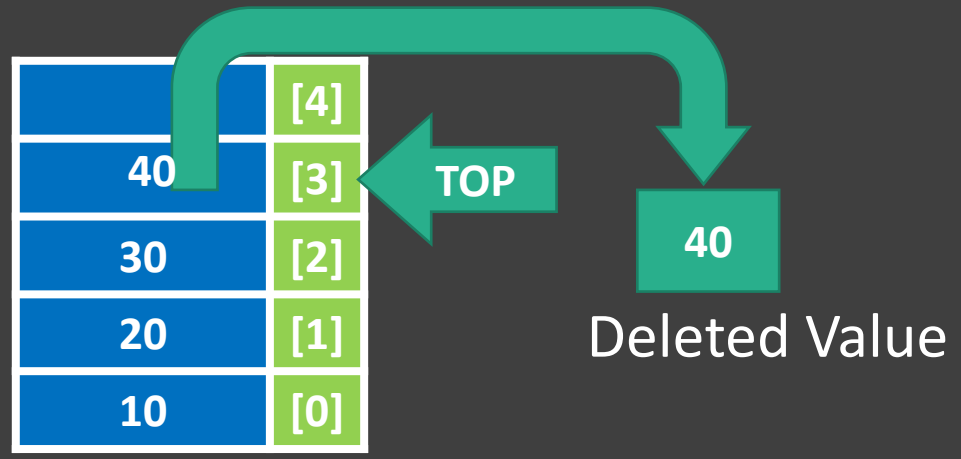
then

Underflow

Step2: Print x is inserted

top- -





DISPLAY( ) Operation:

Using DISPLAY operation we can display all the existing value in the stack

STEPS:

Step1: Check Underflow state of the stack

if(top==-1)

then

Underflow

Step2: for(i=top; i>=0; i- -)

then

print Stack[i]

PEEK( ) Operation:

Using PEEK operation we can display topmost value in the stack

STEPS:

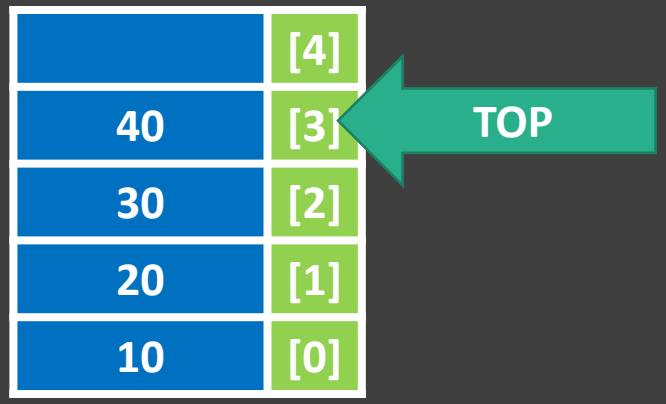
Step1: Check Underflow state of the stack

if(top==-1)

then

Underflow

Step2: print Stack[top] is top most value



COUNT( ) Operation:

Using COUNT operation we can display number of values available in the stack

STEPS:

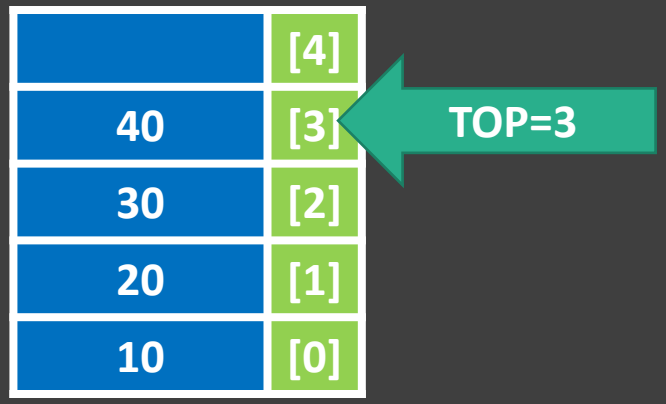
Step1: Check Underflow state of the stack

if(top==-1)

then

Underflow

Step2: print top+1 values are available in the stack



Queue Data Structure – Queue ADT

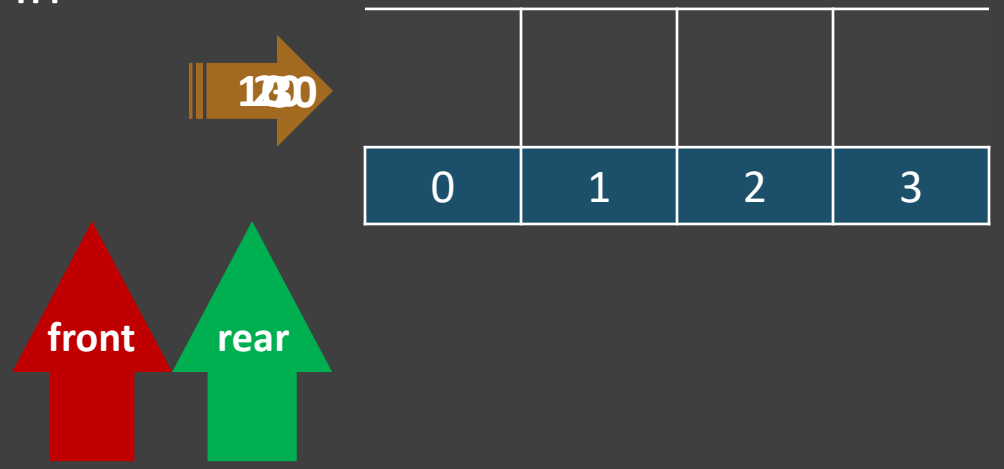
Queue is a linear data structure, that stores values in **continues/sequential** passion.

Queue fallows **FIFO/LILO** strategy

FIFO: First in First Out

LILO: Last in Last Out

Queue is represented Horizontally

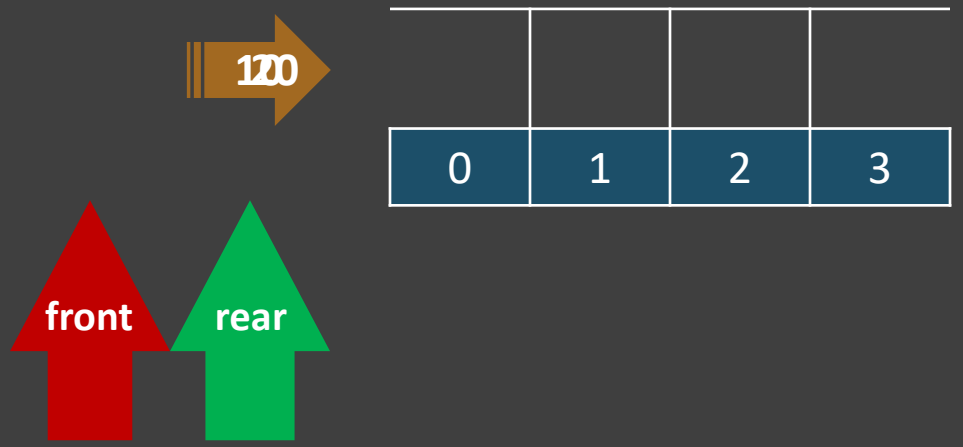


Terminology– Queue ADT

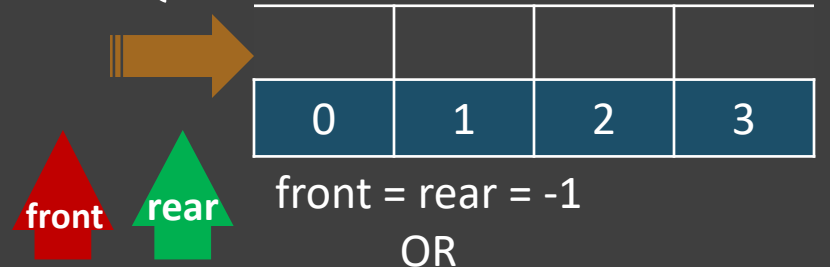
Front: Front is a variable, used to perform delete operations on queue

Rear: Rear is a variable, used to perform insert operations on queue

Default value is **-1** for front and rear variable

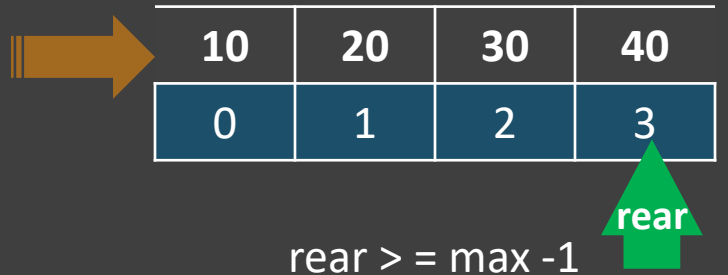


Underflow: It is a state that represents Queue is Empty
We can not **Delete** any value from Queue in Underflow state



front = rear = -1
OR
front == rear

Overflow: It is a state that represents Queue is Full
We can not **Insert** any value from Queue in Underflow state



rear > = max -1

Operations– Queue ADT

1. ENQUEUE(X)
2. DEQUEUE( )
3. DISPLAY( )
4. PEEK( )
5. COUNT( )

Enqueue(x) Operation:

Using this operation we can insert a new value into the Queue

STEPS:

Step1: Check Overflow state of the Queue

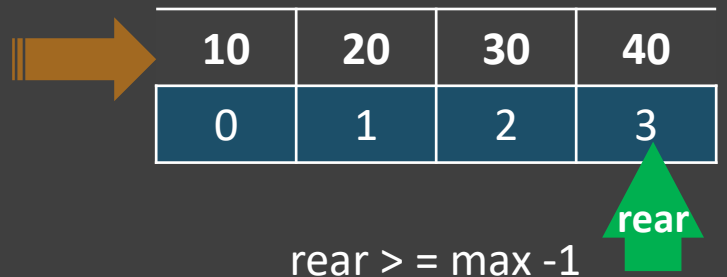
if(rear > = Max -1)

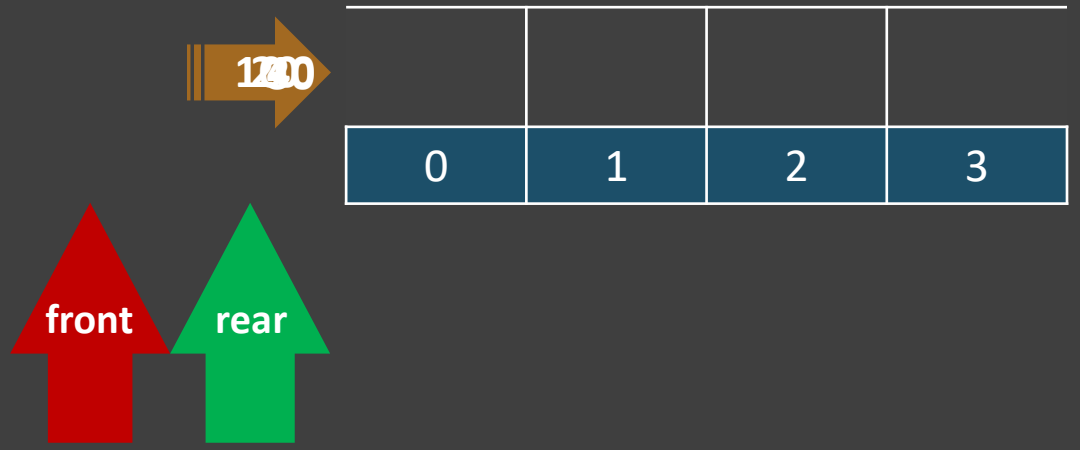
then

Overflow

Step2: rear + +

Queue[rear] = x





Dequeue( ) Operation:

Using this operation we can Delete a value from the Queue

STEPS:

Step1: Check Underflow state of the Queue

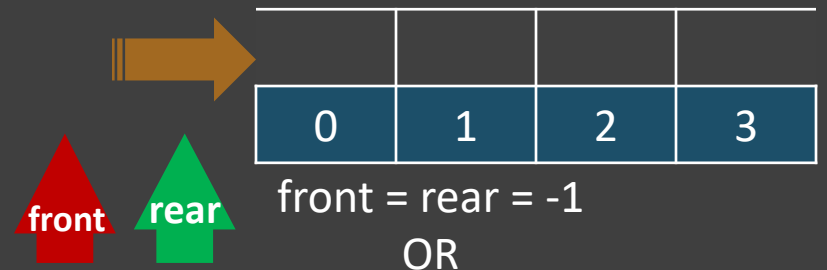
if(front = = rear)

then

Underflow

Step2: print Queue[rear] is deleted

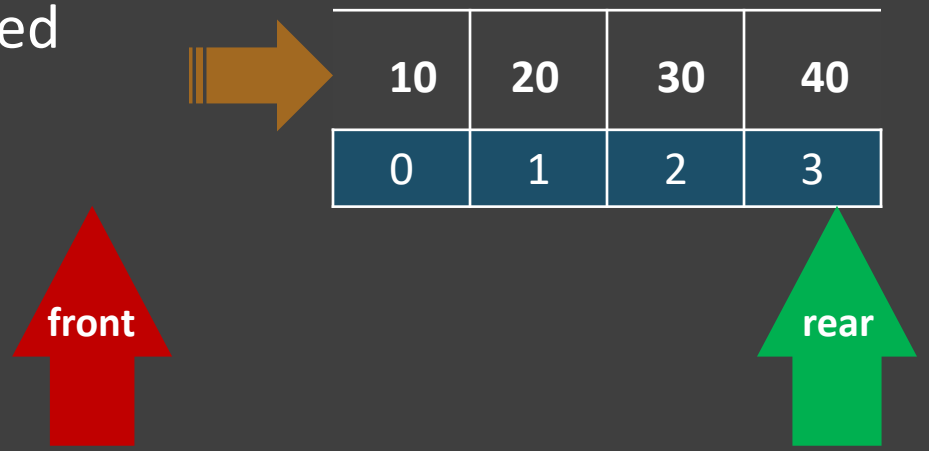
front + +



front = rear = -1

OR

front == rear



Display( ) Operation:

Using this operation we can Display all the values in Queue

STEPS:

Step1: Check Underflow state of the Queue

if(front = = rear)

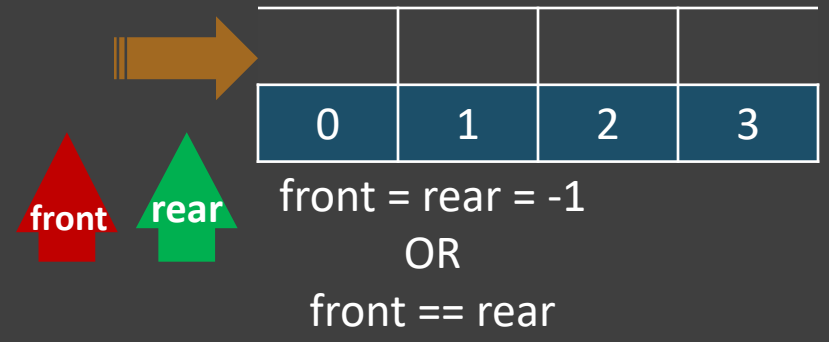
then

Underflow

Step2: for(i= front +1, i <=rear; i++)

then

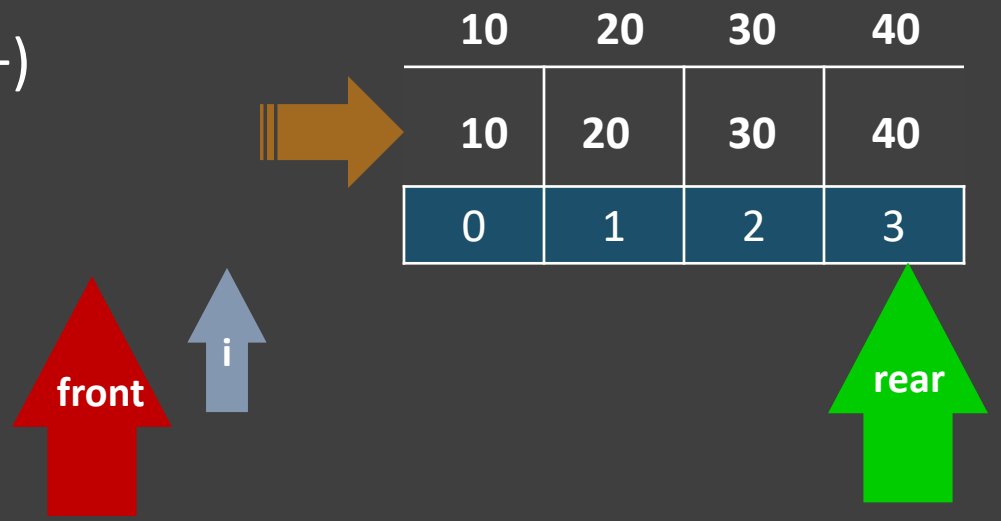
print Queue[i]



front = rear = -1

OR

front == rear



Peek( ) Operation:

Using this operation we can Display the topmost value in Queue

STEPS:

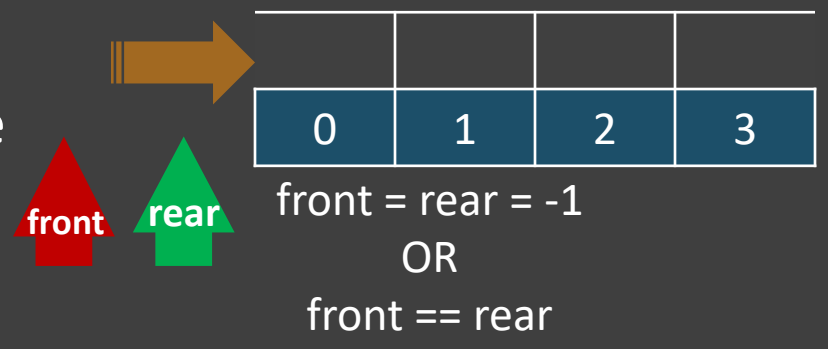
Step1: Check Underflow state of the Queue

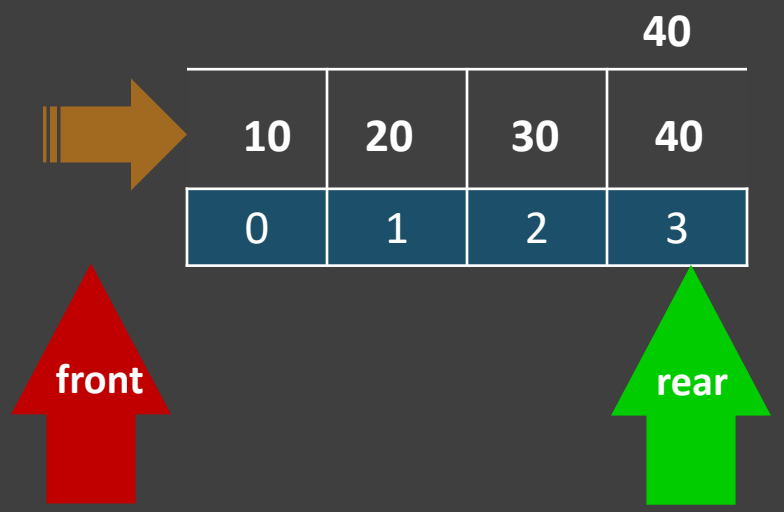
if(front = = rear)

then

Underflow

Step2: print Queue[rear] is topmost value





Count( ) Operation:

Using this operation we can Display the number of values present in Queue

STEPS:

Step1: Check Underflow state of the Queue

if(front = = rear)

then

Underflow

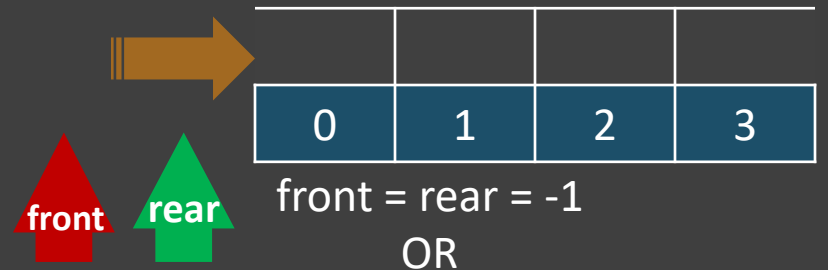
Step2: for(i= front +1, i <=rear; i++)

then

count + +

end loop

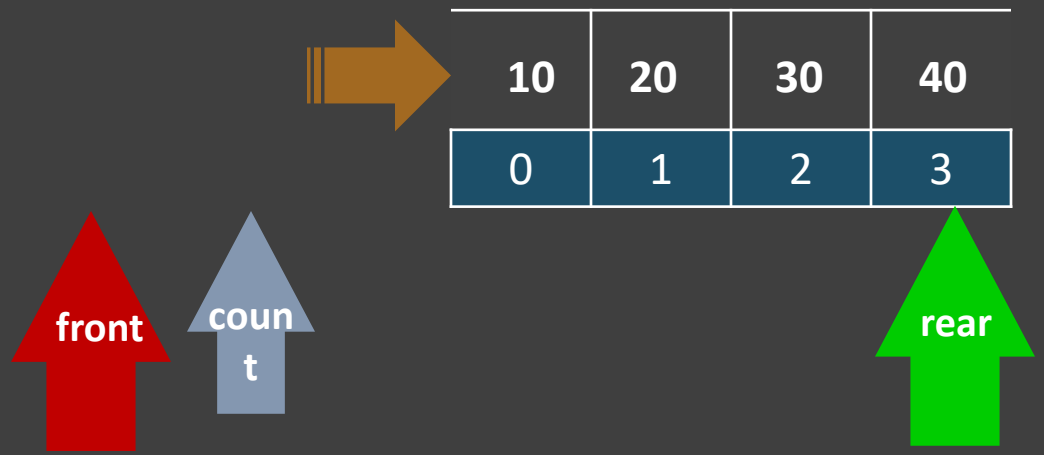
Print count values



front = rear = -1

OR

front == rear



Thank you