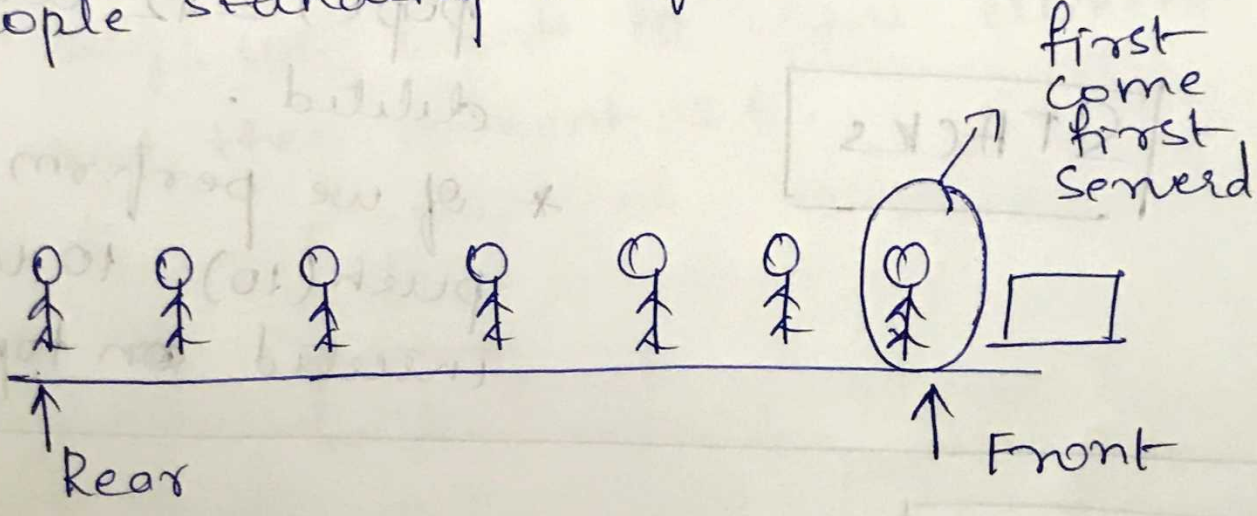
DEFINITION OF QUEUE

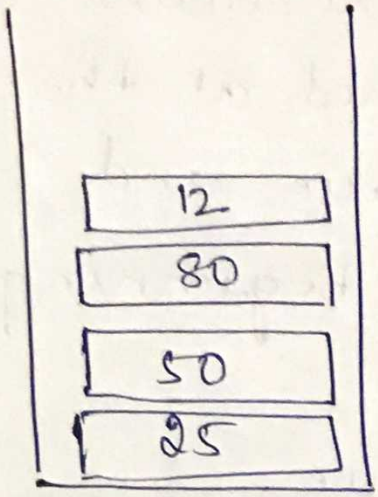
*Queue is a linear data structure in which the insertion is performed at the end called rear of the queue and delition is performed at the beginning called front of the queue.

Ex People standing in queue.



Queue is a FIRST IN FIRST OUT (FIFO) Data structure

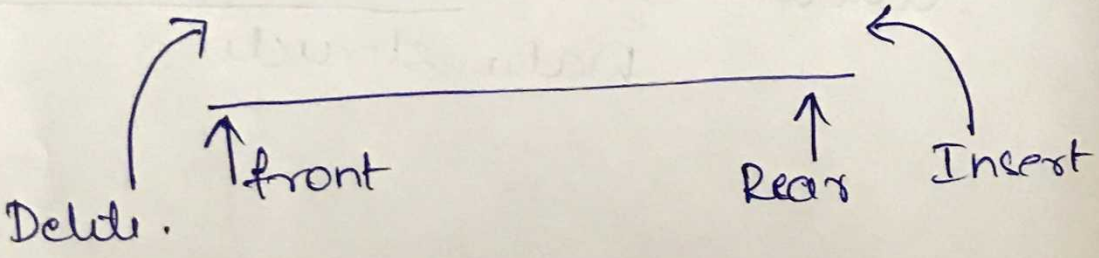
STACKS VS QUEUES



STACKS

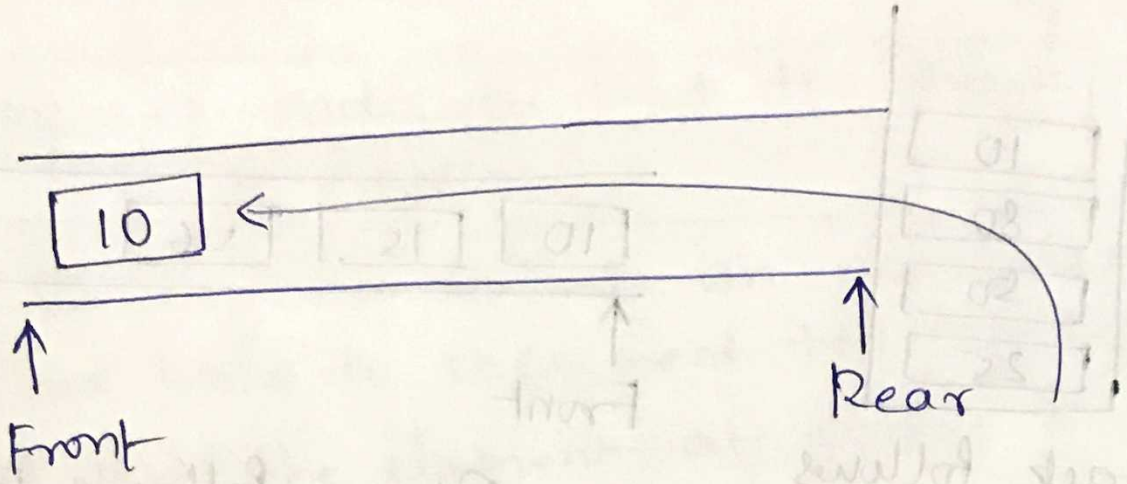
* In case of stack, insertion and deletion is always performed at one end i.e., the top of the stack.
* If we perform pop() = 12 will be deleted.
* If we perform push(10) = 10 will be inserted on top.

QUEUE

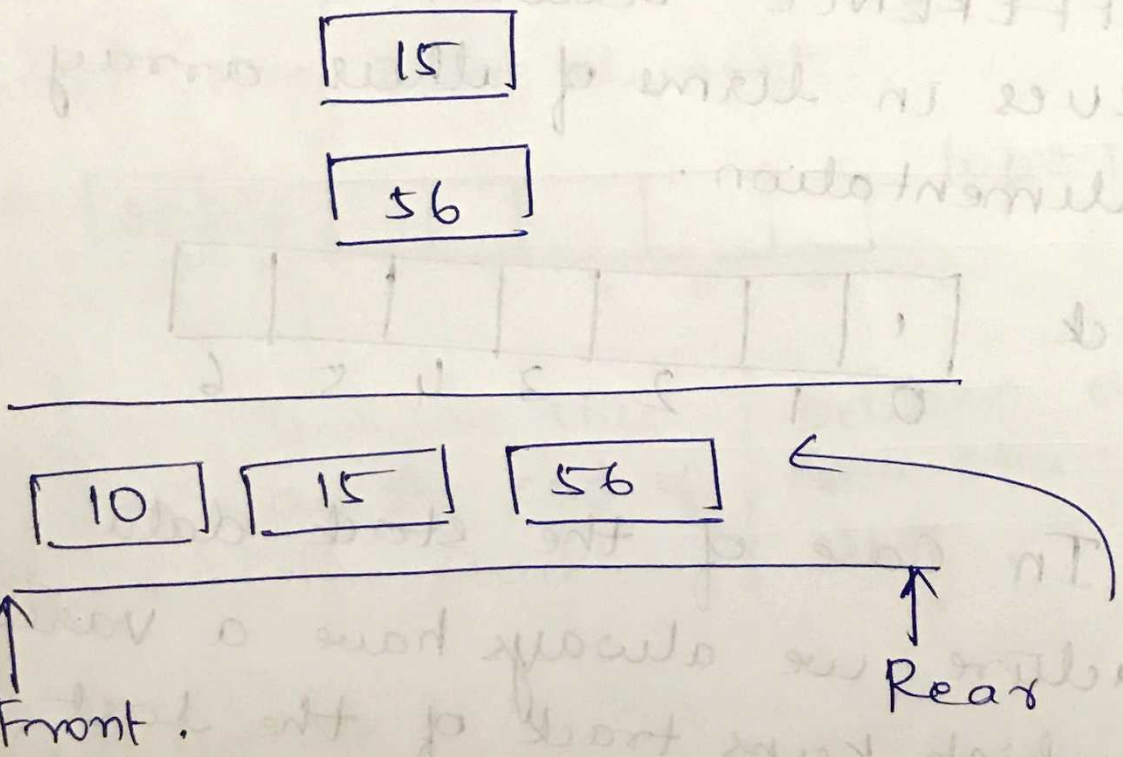


* In case of a queue, insertion is performed at the end and deletion is always performed at the beginning.

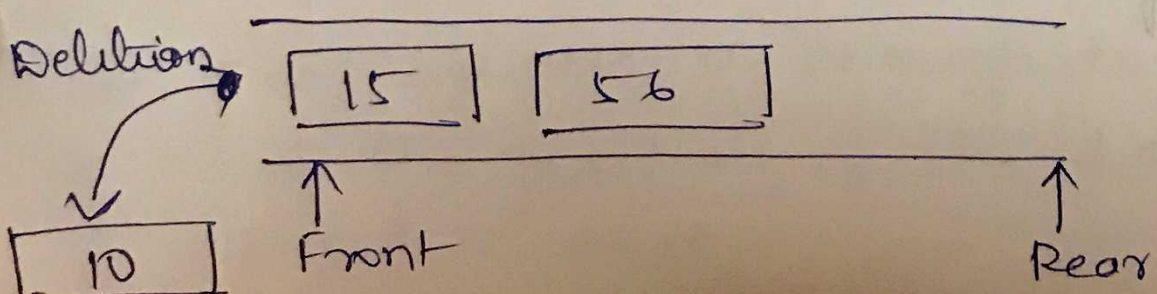
* Let say, we want to insert an element 10



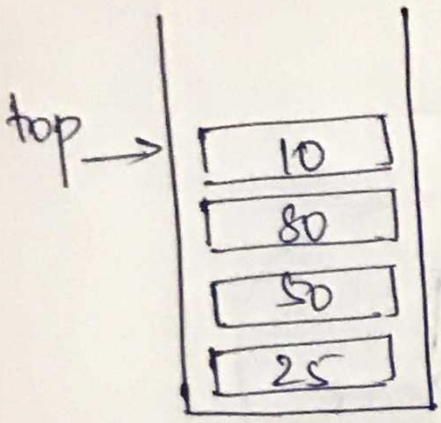
* Let say, we want to insert element 15 and then element 56.



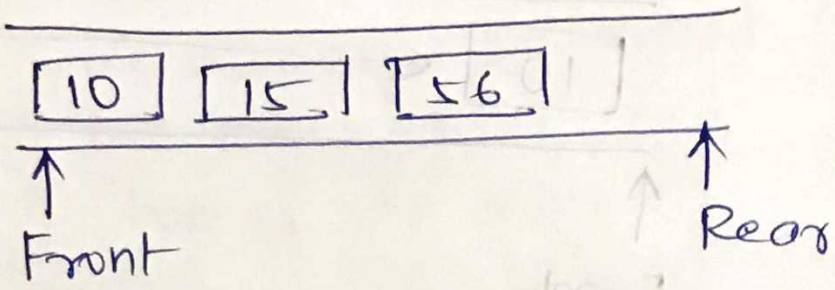
* Let say, we want to delete an element from the queue.



Stack Vs Queues



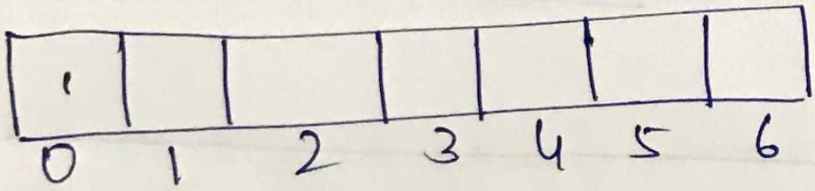
Stack follows LIFO



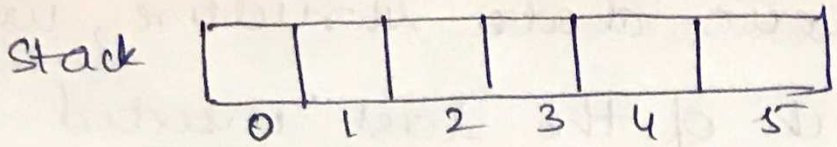
Queue follows FIFO.

* DIFFERENCE between stacks and Queues in terms of their array Implementation.

Stack



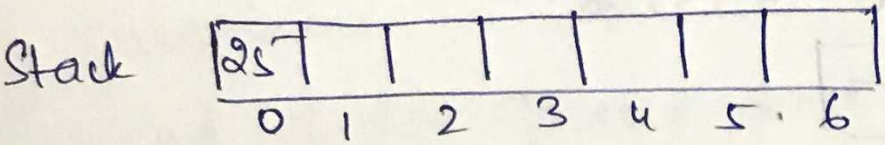
* In case of the stack data structure, we always have a variable top which keeps track of the last inserted element by keeping the index of the last inserted element.



top = -1

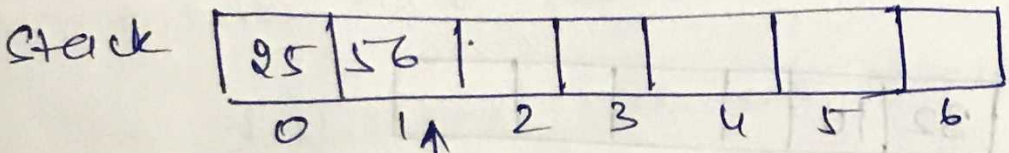
top = -1 indicate that the stack is empty

* If we try to insert an new element, then we have to increment the top by 1 and put the element at index 0.



top = 0.

* Insert one more element.



top = 1

Imagine this as the end of the stack.

Deletion starts from the place itself.

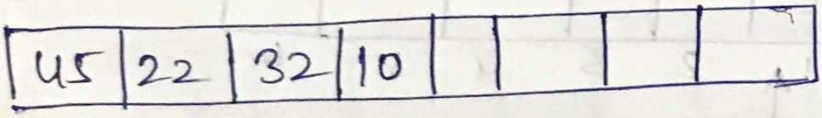
Conclusion

* Insertion and Deletion is allowed only at one end called top of the stack.
* So, one variable is enough to keep track of the position of the last inserted element.

* In case of queue data structure, we have to keep track of the last inserted element and the first inserted element because insertion is performed at the end and deletion is performed at the beginning of the queue.

* we will keep track of the first inserted element through a variable named front.

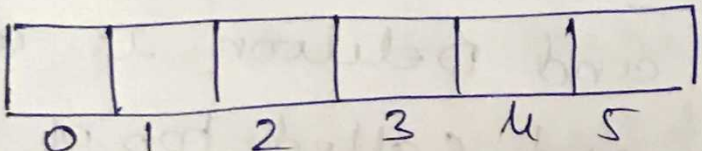
* Consider the current status of the queue.



front = 0
rear = 3

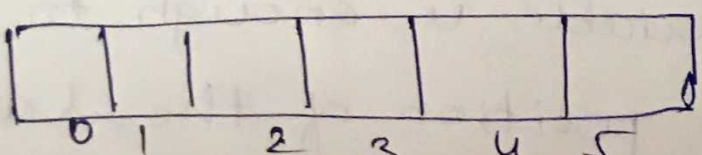
* clear difference between stack and Queue is below.

Stack



top = -1

Queue



front = -1
rear = -1

Important points to remember

* Stack follows LIFO order while queue follows FIFO order.
* Insertion and deletion operations on stack are performed at one end due to which only one variable is enough.
* Insertion in queue is performed at the end of the queue and deletion at the front of the queue due to which minimum two variables are required.
* The diffirence between stacks and Queues lies in their removal procedure. The most recently added element will be deleted in stacks while the least recently added element will be deleted in queues.

Array Implementation of Queues

Problem statement :- Write a program to implement the queue data structure using arrays.

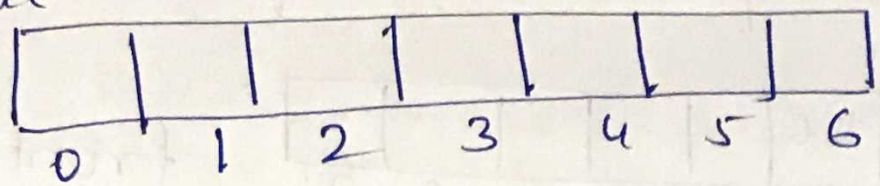
* Let's do the requirement analysis first.

TODO list-

* Add Preliminaries and create the prompt for the user.
  Ex Header file, Variables and so on.
* Define the enqueue() function.
* Define the dequeue() function.
* Define the isEmpty() and isFull() function.
* Define the print() function.

enqueue() : Insert an element in the queue.

queue



front = -1
rear = -1

* Algorithm :

1. Increment the rear by 1
2. Put the new Element at index specified by rear.

* Enqueue (8).
  1. rear = rear + 1
  2. queue[rear] = 8.
* Enqueue (65).
  1. rear = rear + 1
  2. queue[rear] = 65.

front has to increment.

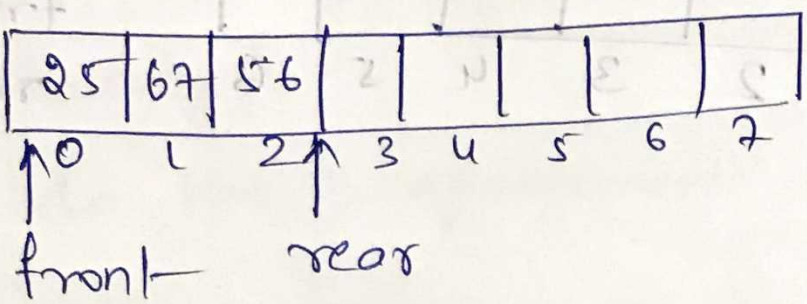
* updated algorithm :

1. If front equal to -1, then increment front by 1.
2. Increment the rear by 1
3. Put the new element at index specified by rear.

dequeue() : delete an element from the queue.

* Consider the present array (queue)

queue



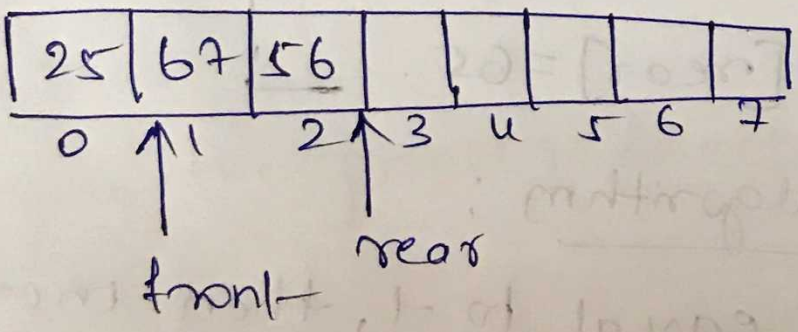
front = 0
rear = 2

* Algorithm

1. Store the first element in some variable (let say, variable name is data).
2. Increment the front by 1
3. Return the data.

* dequeue

queue



front = 1
rear = 2

* The element 25 is ignored and not removed from the array.