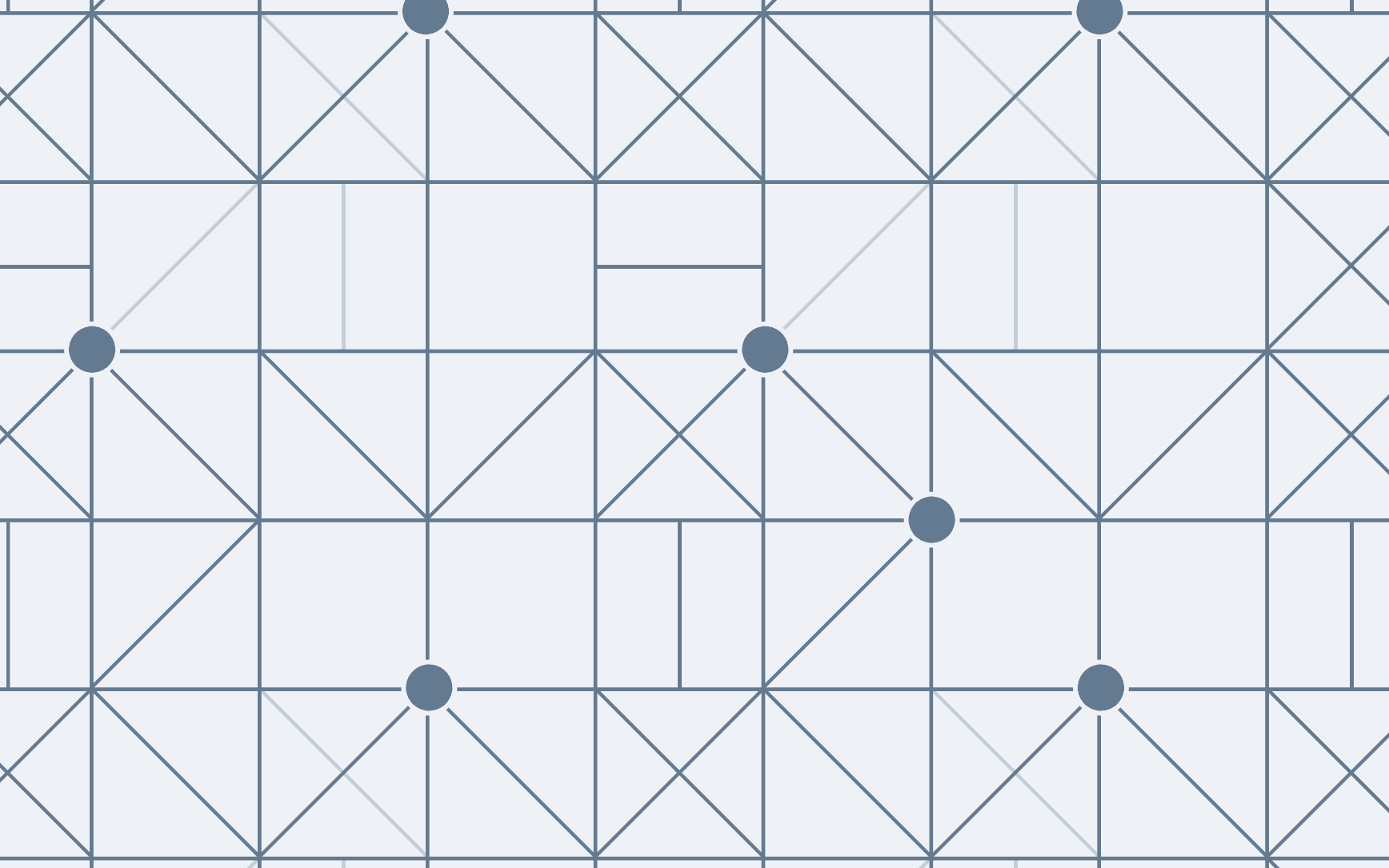


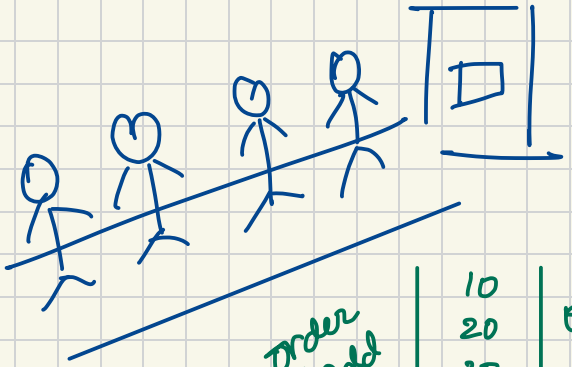
Queues.

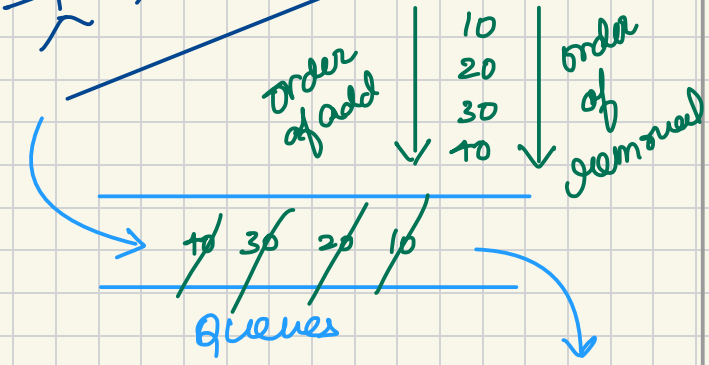
Agenda

1. Intro. about Queues
2. Queues and Stack Easy Problems
3. 2 Stack using an Array
4. Implementing Q que using Stack

Queues

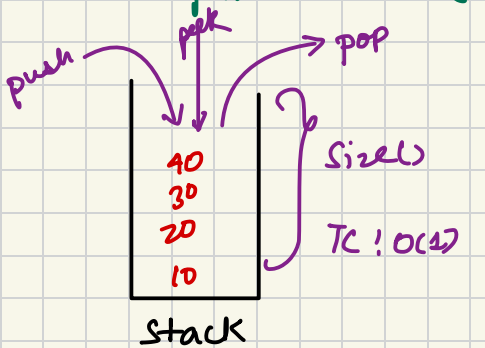
- Linear DS
- FIFO
  - (first in, first out)

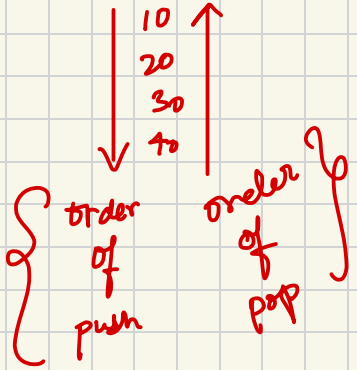


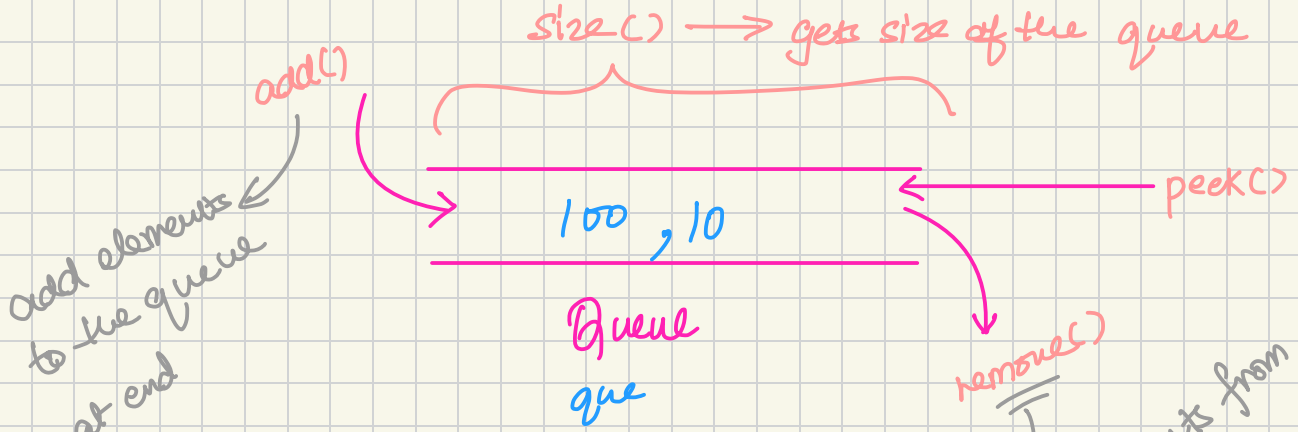


Stacks

- Linear DS
- follows LIFO



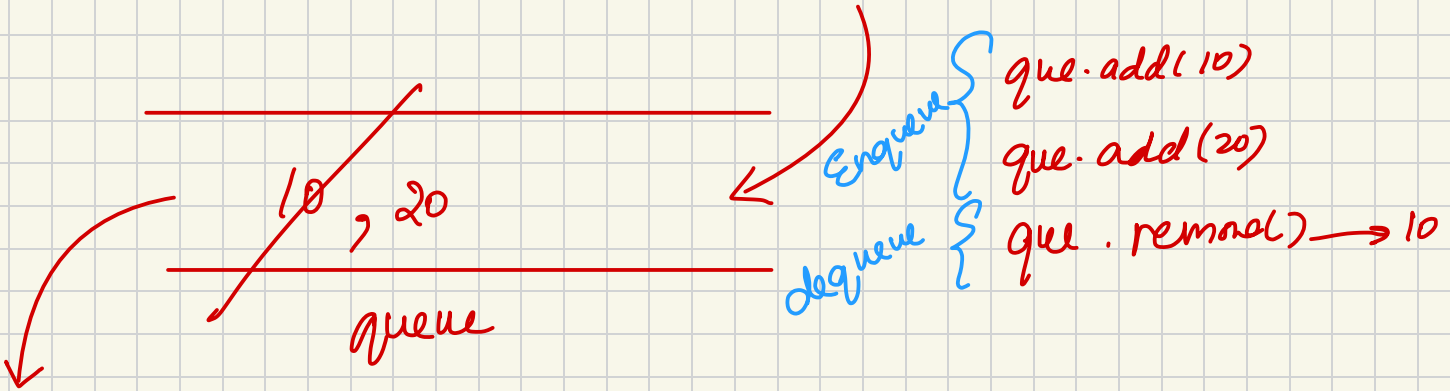





```
que.add(90)
que.add(10)
que.size() ⇝ 2
que.peek() ⇝ 90
que.remove() ⇝ 90
que.add(100)
```

Enqueue. { Enter in queue } add()

Dequeue. { Delete from queue } remove()



Queues

- → Linear data structure
- → follows first in, first out

Methods

1. add()
2. remove()
3. peek()
4. size()

TC: O(1)
SC: O(1)

1. offer();
2. poll();
3. peek();
4. size();

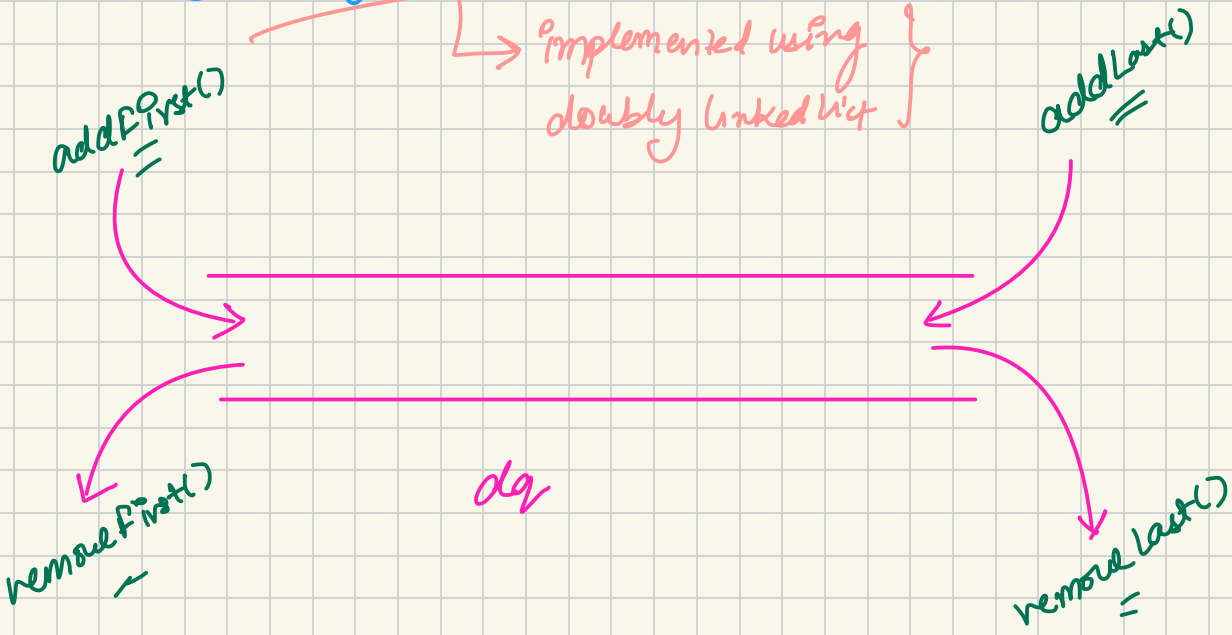
```
Queue<G> que_name = new ArrayDeque();
                  = new LinkedList();
```

Deque

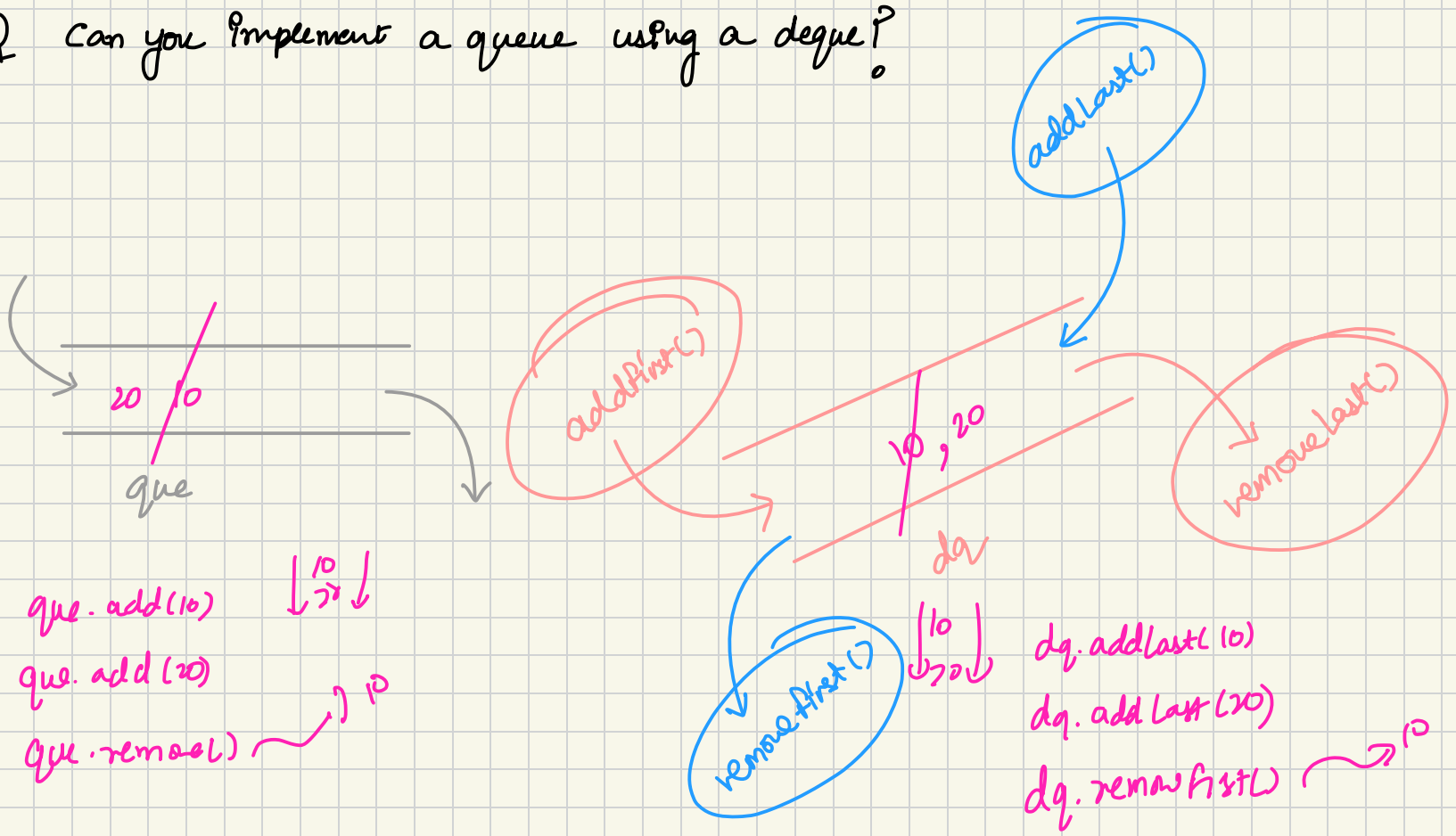
{ doubly Ended Queue }

→ implemented using doubly linked list

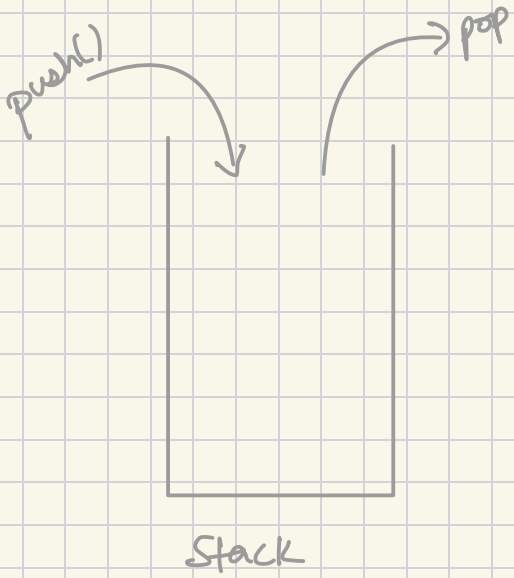
Linear DS

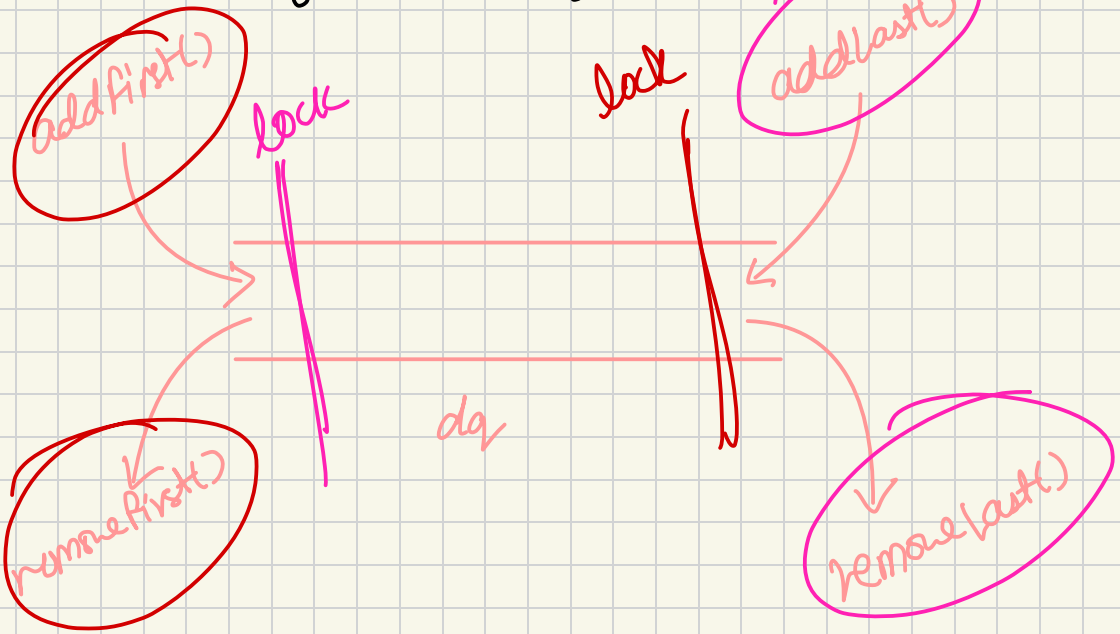


Q Can you implement a queue using a deque?



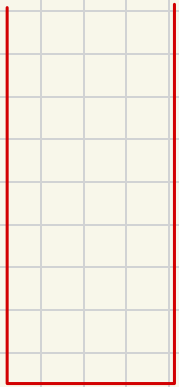
Q Can you implement a stack using a deque?

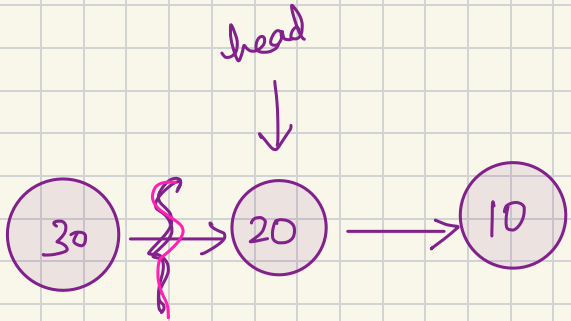




Design a Stack using LinkedList.

- ✓ push(10)
- ✓ push(20)
- ✓ push(30)
- peek() → 30
- pop() → 30
- peek() → 20
- push(40)
- pop() → 40
- pop() → 10

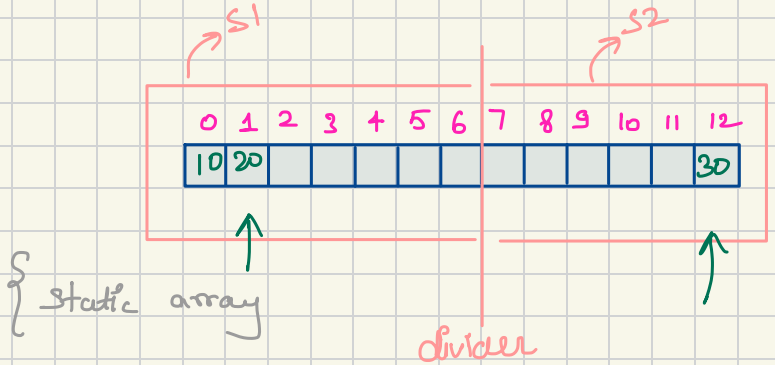




node.next = head

{ addFirst() → LinkedList (push)
  removeFirst() → LinkedList (pop) }

Implement 2 Stacks using an array ! → Amazon.

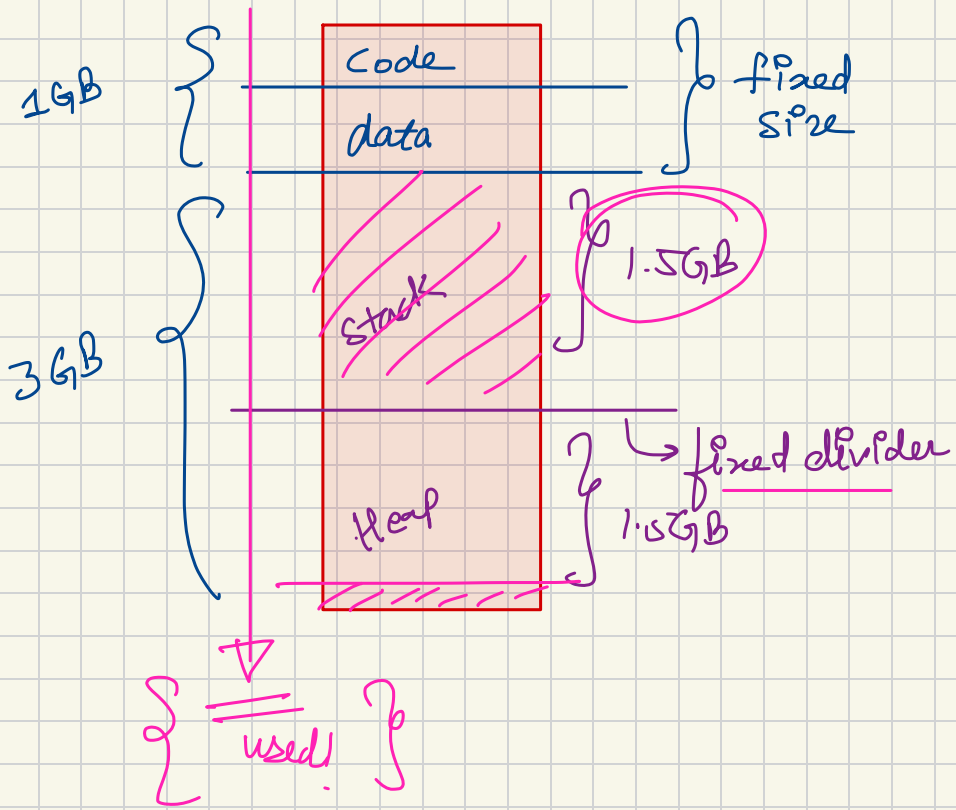


s1. push(10)
s1. push(20)
s2. push(30)

OS.

4GB

Memory.

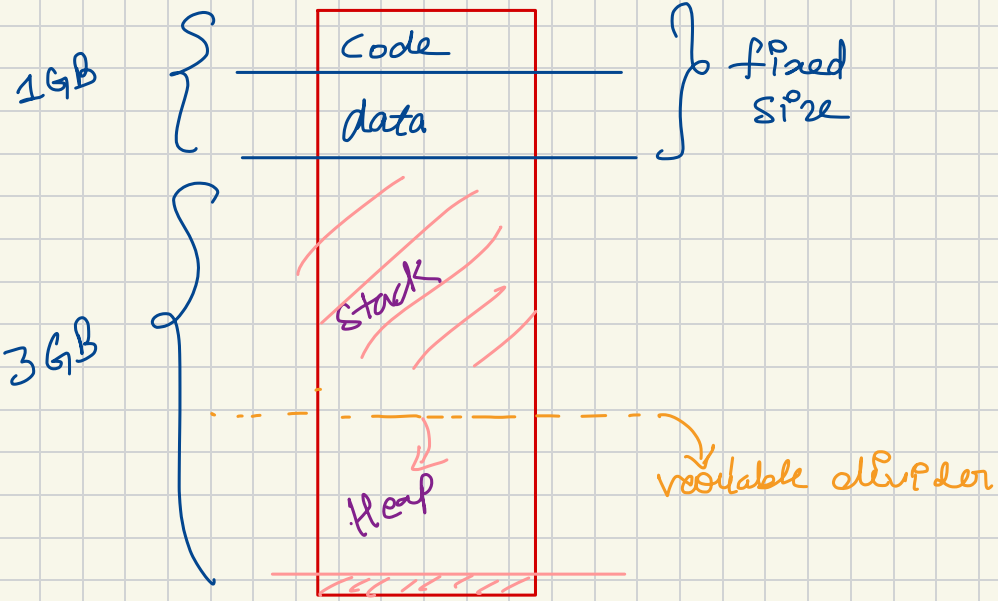


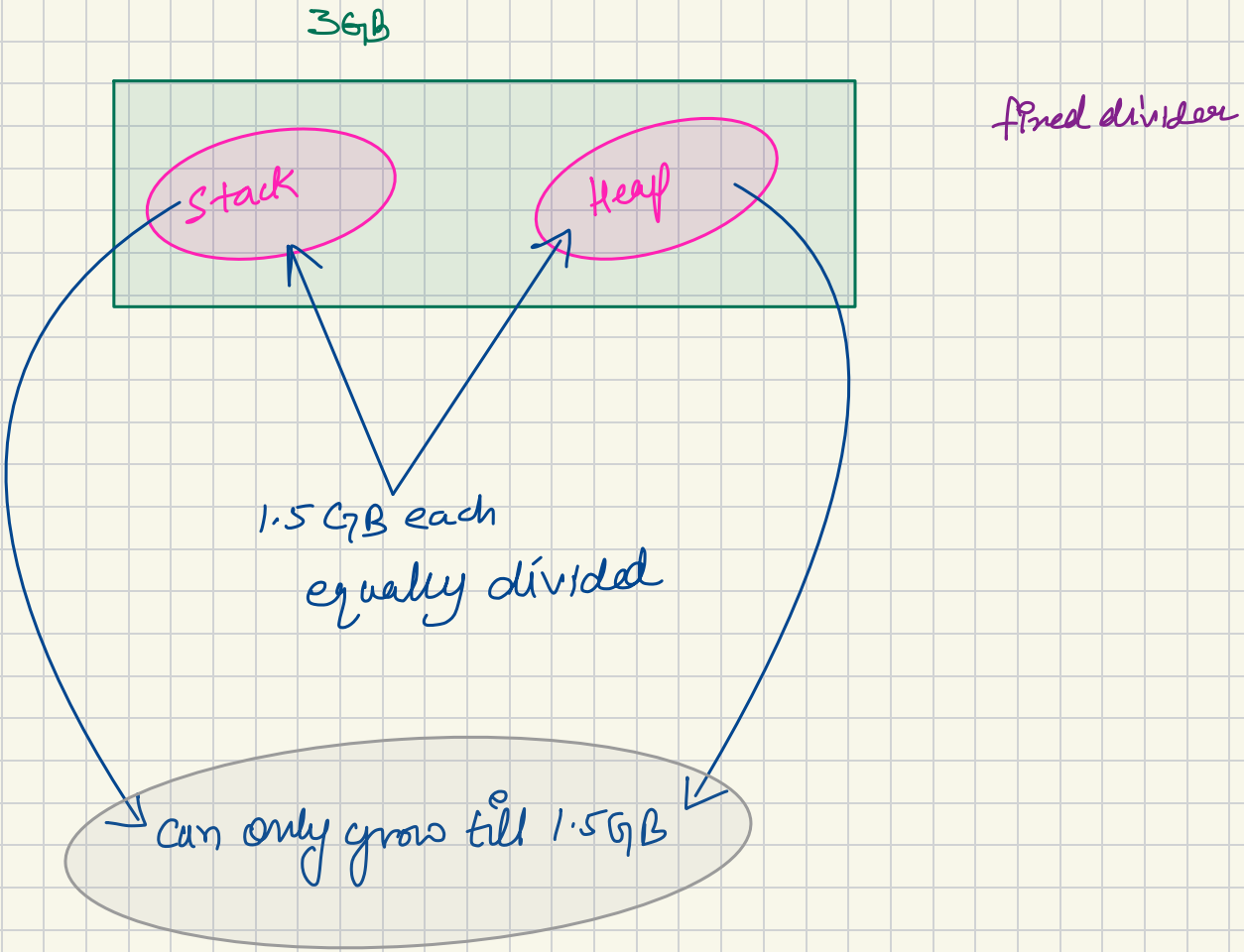
Crash

OS

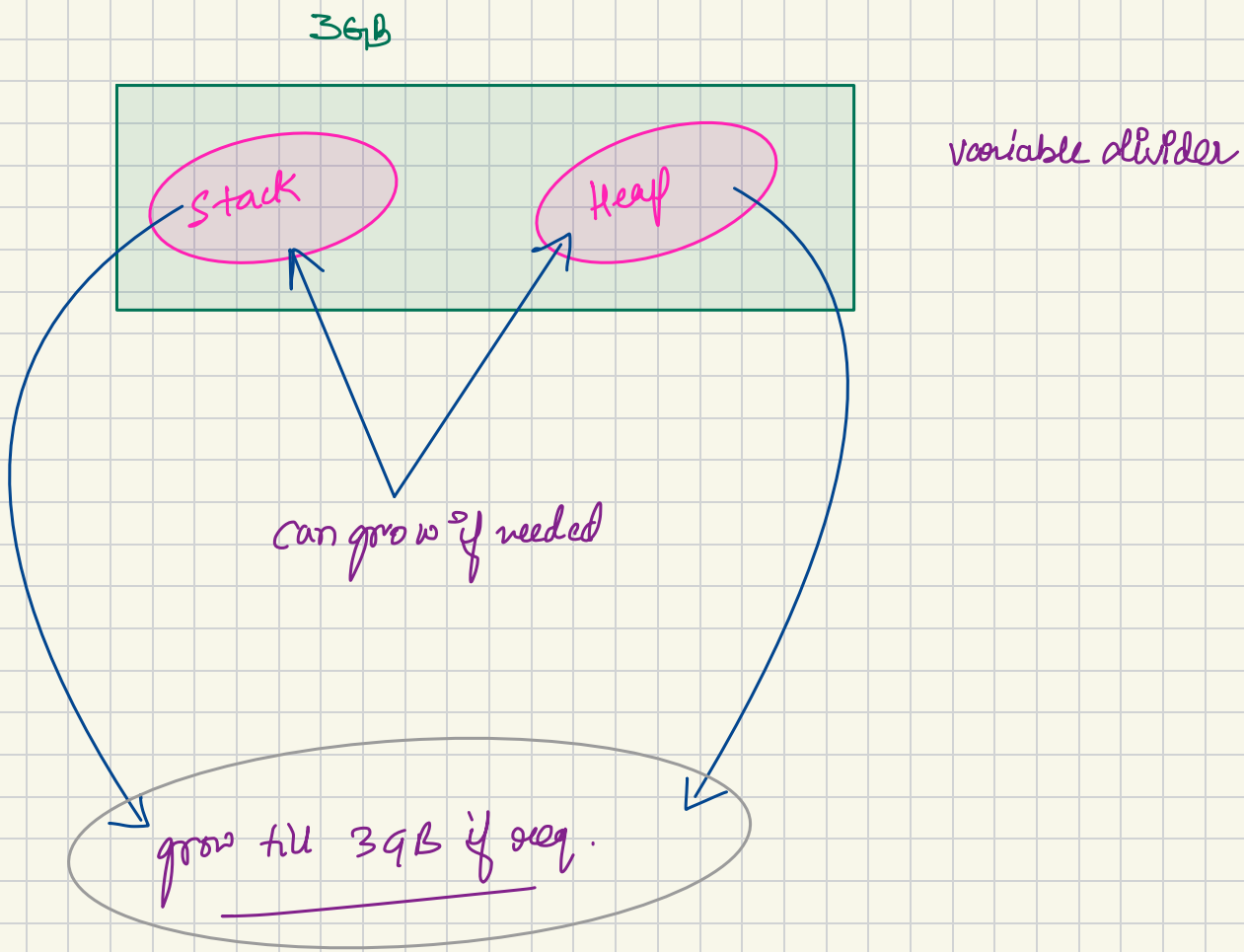
4GB

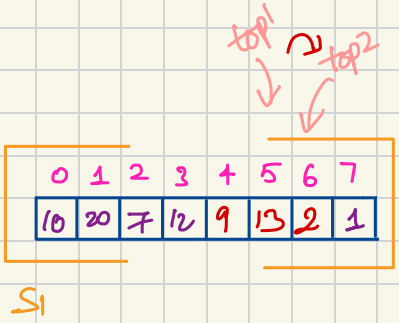
Memory.





fixed divider


3GB
Stack
Heap
variable divider
can grow if needed
grow till 3GB if req.



① S1.push(10)

② S1.push(20)

③ S2.push(1)

④ S1.push(7)

⑤ S1.push(12)

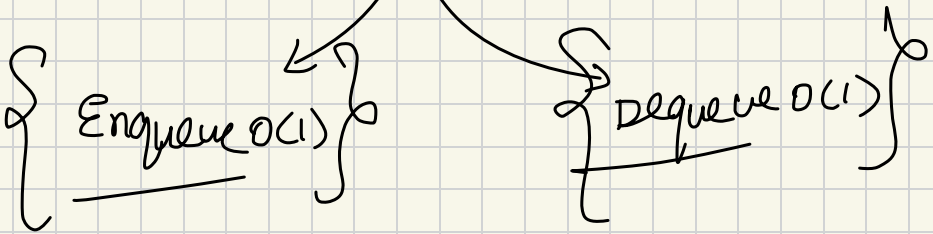
⑥ S1.push(9)

⑦ S2.push(2)

⑧ S1.push(10)

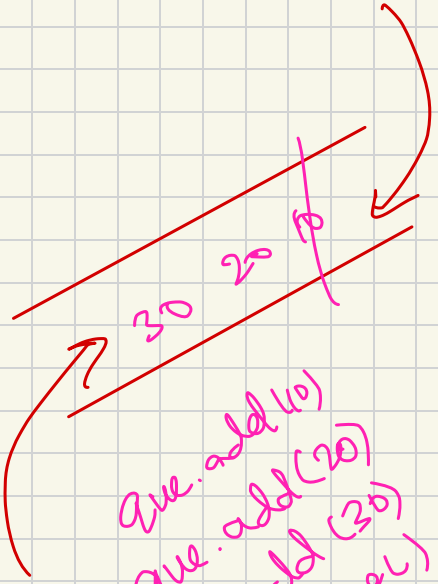
⑨ S1.push(20)

Implement Queue using 2 stacks.

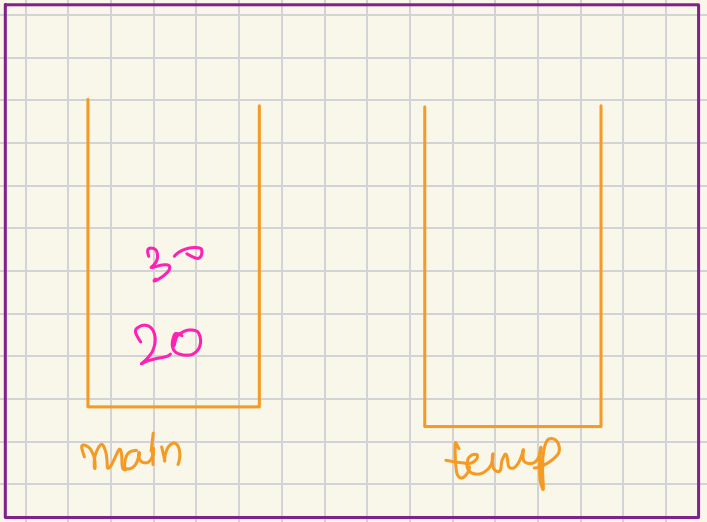


Implement Queue using 2 stacks such enqueue O(1)

10



que.add(10)
que.add(20)
que.add(30)
que.remove()



enqueue()
main.push(x) } O(1)

dequeue
↳ O(N)