

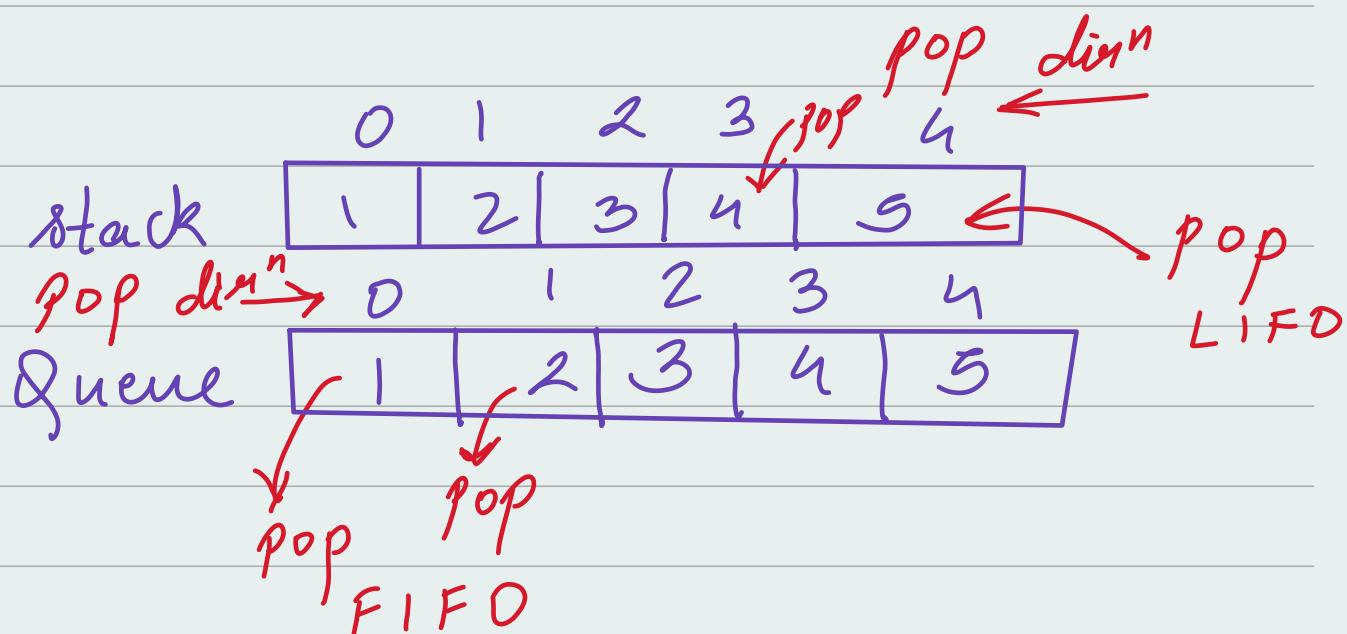
Questions on Stacks 3

Queues

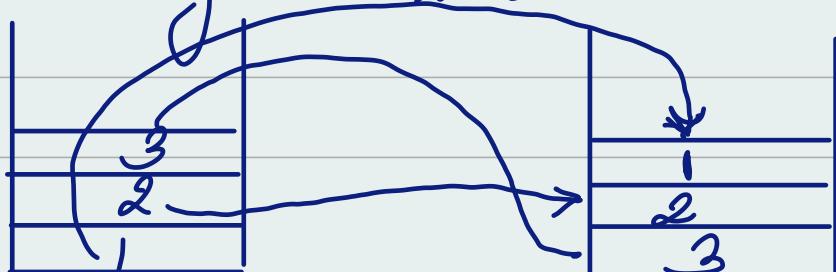
b) Implement a Queue using stacks.

In Queue ! First in first out

So to implement like queue using stacks, we will use two stack first stack & second stack.



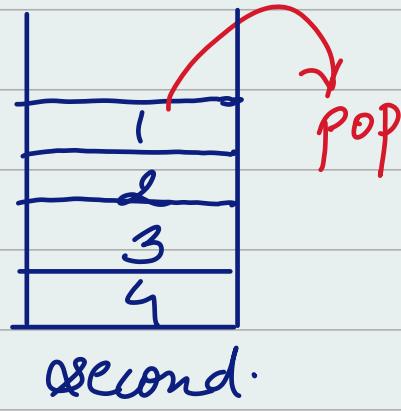
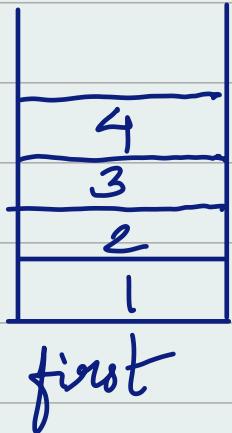
So, Using two stacks



first stack

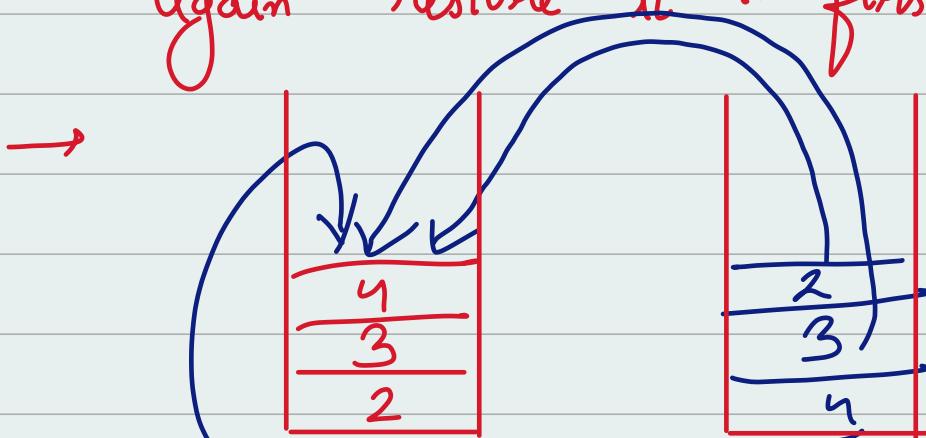
Second stack

→ Empty the first stack into the second.



→ first remove or pop the elements from the first stack into second stack

→ Remove ① from second stack and then again restore it to first stack

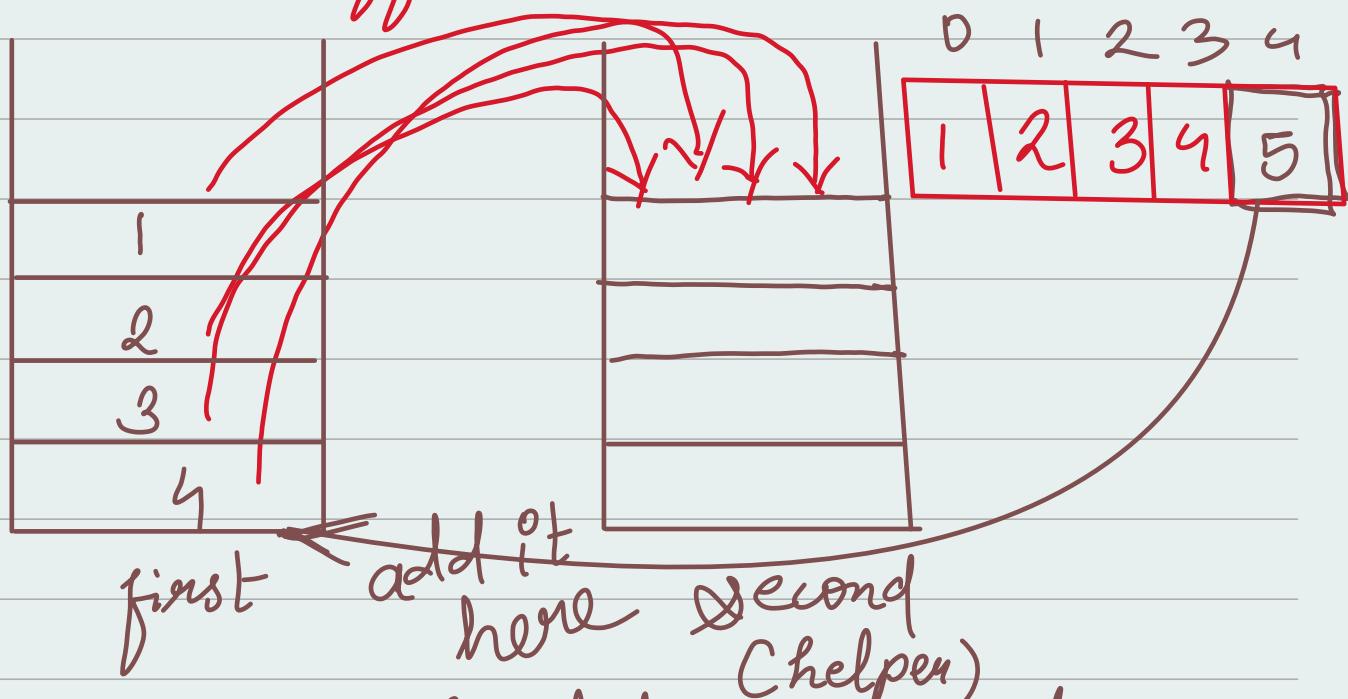


Second pop into first stack.

→ Now it is behaving like queue' (FIFO)

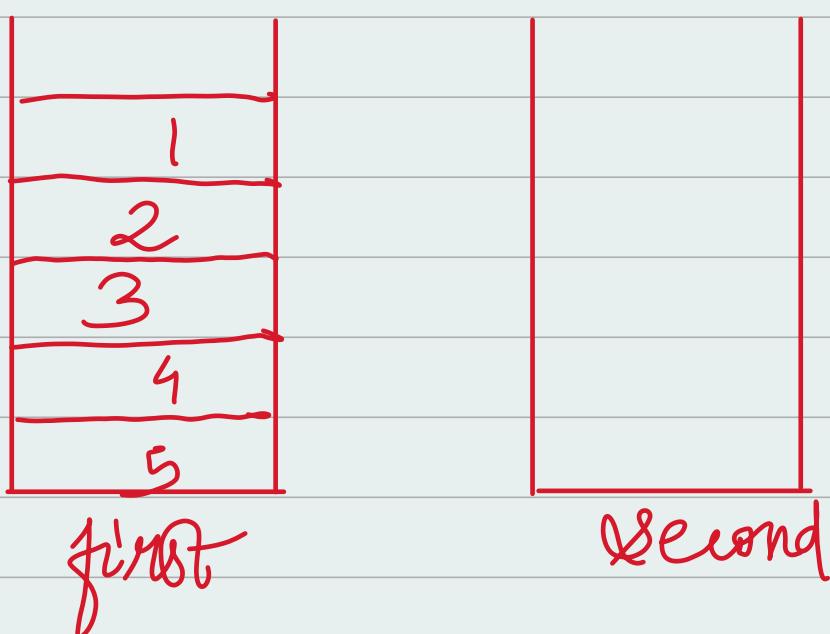
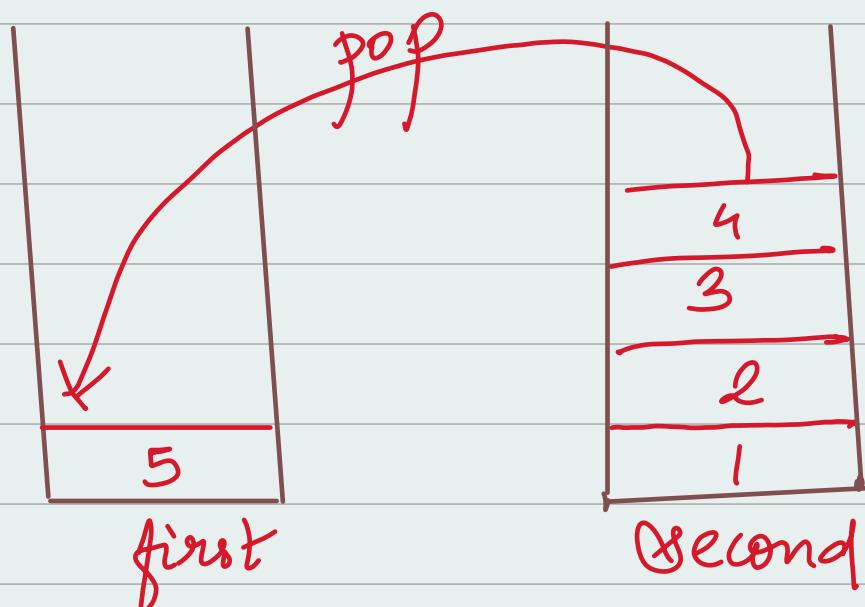
→ Complexity is $O(N)$ for removing the elements from first stack and adding it into second stack. The complexity of adding elements are $O(1)$. Similarly from second stack to first stack (complexity) $O(N)$ for removing all the elements from 2 to n but adding it into the first stack is $O(1)$.

→ Remove efficient



Here we inserted the numbers in reverse list manner. But the problem is if we have to add 5 to the list we directly cannot do that for that we have to use helper function that is second stack. We will

pop all the elements from first stack to second stack and then add 5 into the first stack. After that removing or popping out the elements again from second stack to first stack. In this way we can add an elements in the end of the stack.



// Here O(1) is removal &
// O(N) is insertion time.

Code :

class QueueUsingStack {

private Stack < Integer > first;

private Stack < Integer > second;

public QueueUsingStack () {

first = new Stack <> ();

second = new Stack <> ();

}

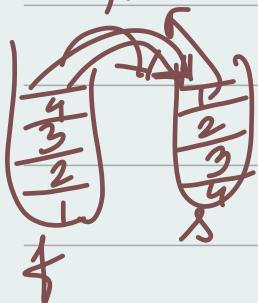
public void add (int item) {

first.push (item);

}

public int remove () throws Exception {

removed while (! first.isEmpty ()) {



} second.push (first.pop());

```
int removed = second.pop(); //  
while (!second.isEmpty()) {  
    first.push(second.pop());  
}  
return removed;
```

```
}  
public int peek() throws Exception {  
    while (!first.isEmpty()) {  
        second.push(first.pop());  
    }
```

```
int peeked = second.peek(); //  
while (!second.isEmpty()) {  
    first.push(second.pop());  
}  
return peeked;
```

```
}  
public boolean isEmpty() {  
    return first.isEmpty();  
}
```

3

Code : Queue Using Stack remove

class QueueUsingStack {

private Stack <Integer> first ;
private Stack <Integer> second ;

public QueueUsingStack () {

first = new Stack <> () ;

second = new Stack <> () ;

}

public void add (int item) throws Exception { // Adding S into
while (!first.isEmpty ()) { the list
 second.push (first.pop ()) ;

 first.push (item) // S number is pushed into
 while (!second.isEmpty ()) { first stack
 first.push (second.pop ()) ;

public int remove () throws Exception

 while (!first.isEmpty ()) {

 second.push (first.pop ()) ;

```
int removed = second.pop(); // 1  
while (!second.isEmpty()) {  
    first.push(second.pop());  
}  
}  
return removed;
```

```
}
```

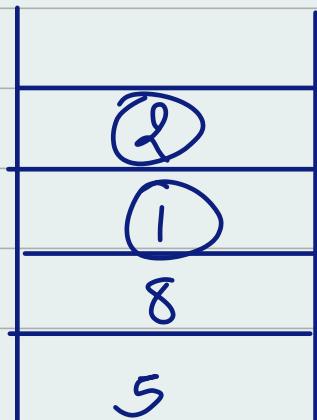
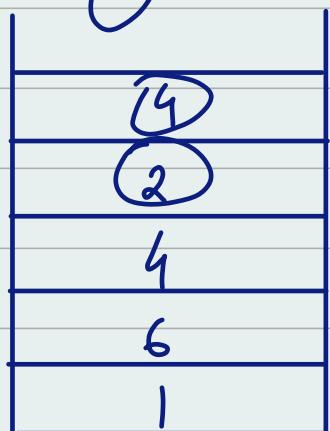
```
public int peek() throws Exception {  
    return first.peek();  
}
```

```
}
```

```
public boolean isEmpty() {  
    return first.isEmpty();  
}
```

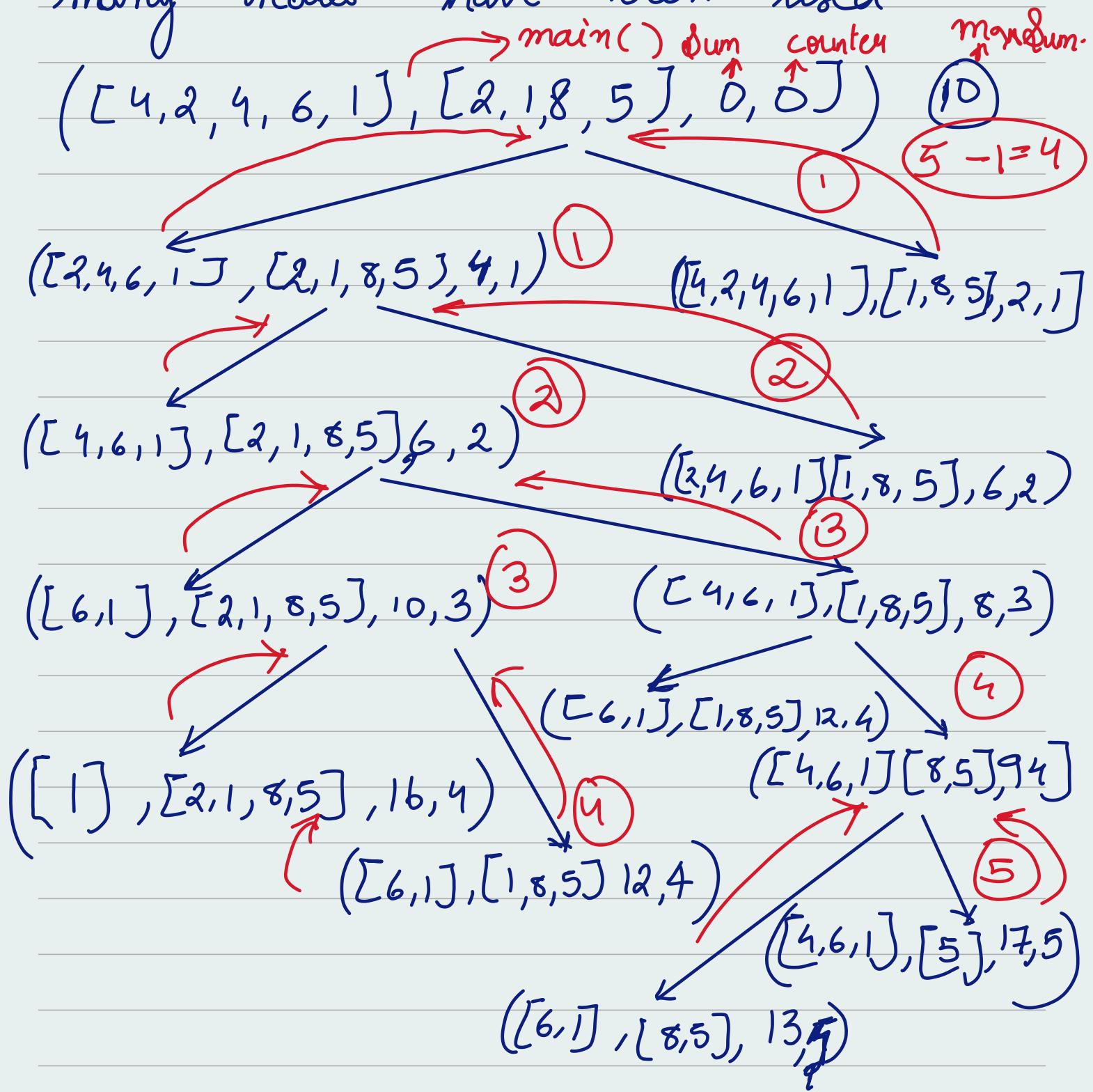
```
}
```

Q) Game of Two Stacks ~ Hacker Rank Problem.



Take stacks sum such as it should not go more than $\text{maxSum} = 10$

For example : $(4+2+2+1) = 9$
and take a counter like how many nodes have been used.



The 5th counter is breaking the condition & so we have to subtract with 1 $\Rightarrow 5 - 1 = \underline{\underline{4}}$ ans

Code :

class TwoStacks {

static int twoStacks (int x, int []a,
int []b) {

return twoStacks (x, a, b, 0, 0) - 1;

}

private static int twoStacks (int x,
int []a, int []b, int sum,
int count) {

if (sum > x) {

return count;

}

if (a.length == 0 || b.length == 0) {

return count;

}

`int ans1 = twoStacks(x, Arrays.
copyOfRange(a, l, a.length), b, sum + a[0],
[4 2 4 6] count + 1);`

$\begin{matrix} & \downarrow \\ 2, 4, 6, 1 \end{matrix}$

`int ans2 = twoStacks(x, a, Arrays.
copyOfRange(b, l, b.length), sum + b[0], count + 1);`

$\begin{matrix} & \downarrow \\ 1, 8, 5 \end{matrix}$

`return Math.max(ans1, ans2);`

3

`public static void main(String[],
args) {`

`Scanner s = new Scanner(System.
in);`

`int t = s.nextInt(); // Taking
number of test cases`

`for (int i = 0; i < t; i++) {
// Running the loop for that many test cases times`

`Taking // int n = s.nextInt();
size of a`

Taking size
of b //int m = s.nextInt();
Taking sum//int x = s.nextInt();

int [] a = new int [n];
int [] b = new int [m];

for (int j=0; j < n; j++) {

a[j] = s.nextInt();

}

for (int j=0; j < m; j++) {

b[j] = s.nextInt();

}

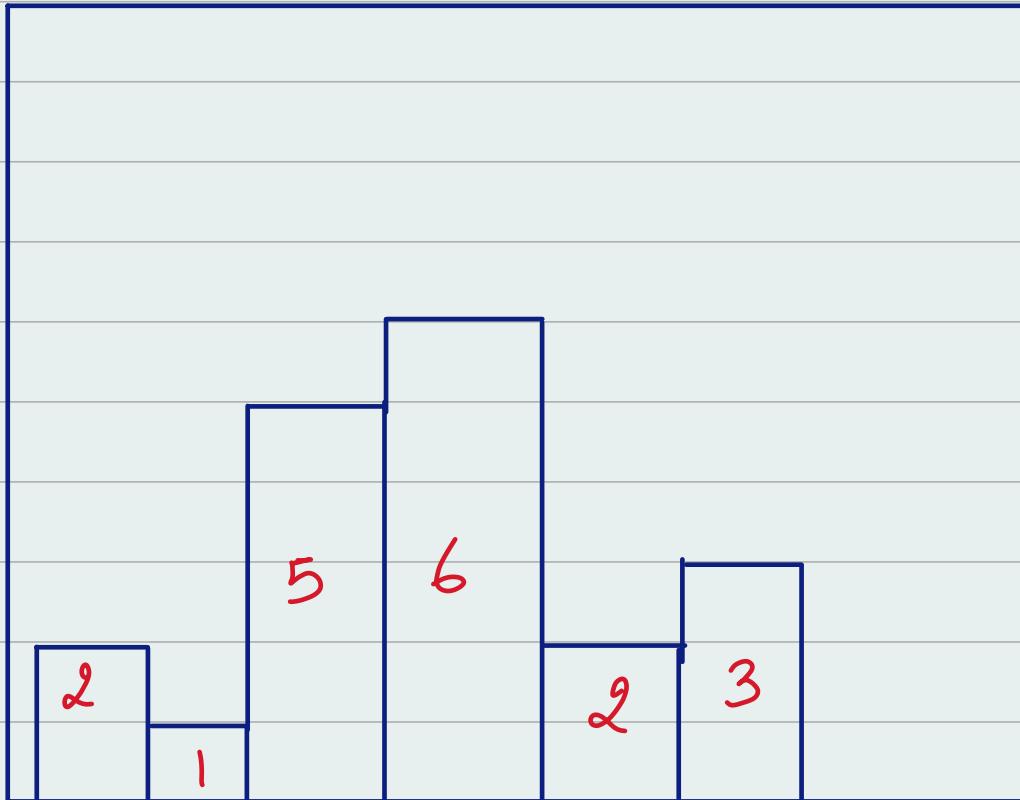
System.out.println (twoStacks
(x, a, b));

3

3

3

D) Find the largest area in histogram? ^(rectangle)

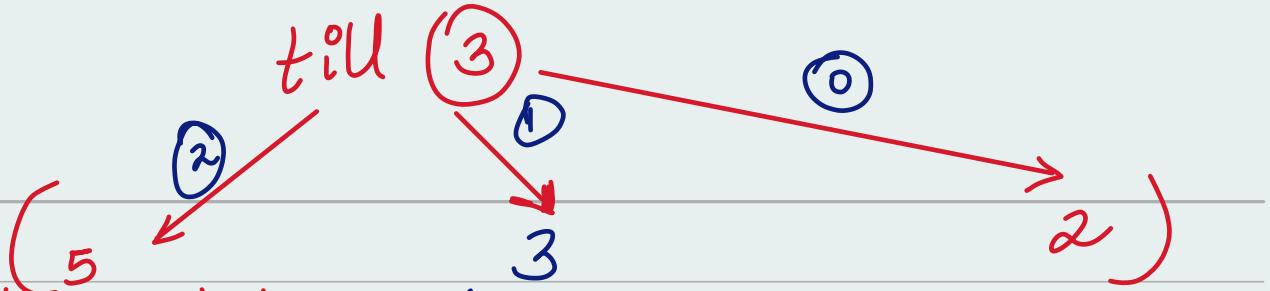


index → 0 1 2 3 4 5

* max till each index.

till $\textcircled{1} \rightarrow 2$ (means 0th index value)

till $\textcircled{2} \rightarrow 2$ (1^{st} index connected to 0th index
by making it 2 so we can move left)
max = 2
2 {The value of index 0}
so the maximum of these two values would be 2



(We cannot move right & we cannot move left cause the 2nd index is not properly connected to 1st index)

(Means the bar 5 is not properly connected to 1.)

(We can move

right as well as

left cause bar 1 is conn' to

is properly conn' to index 0 and 2)

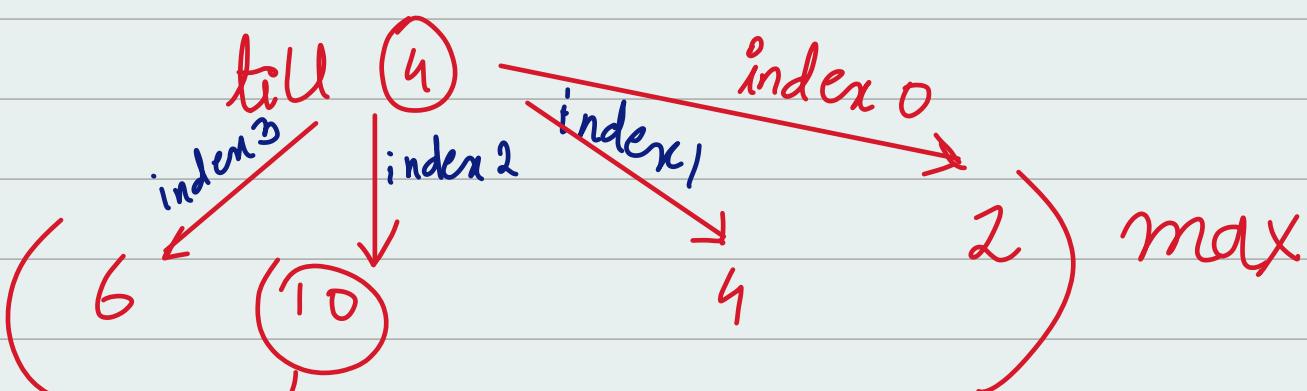
(Here the bar 2

is conn' to 1 but not

properly.

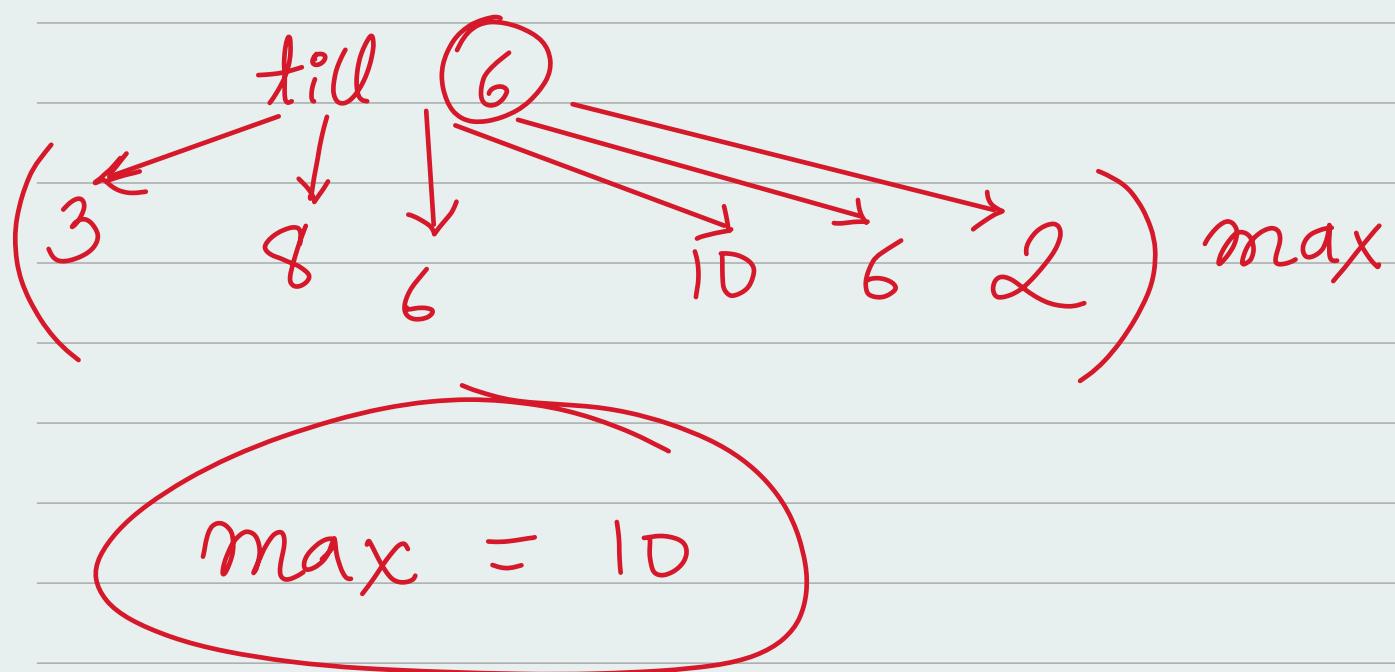
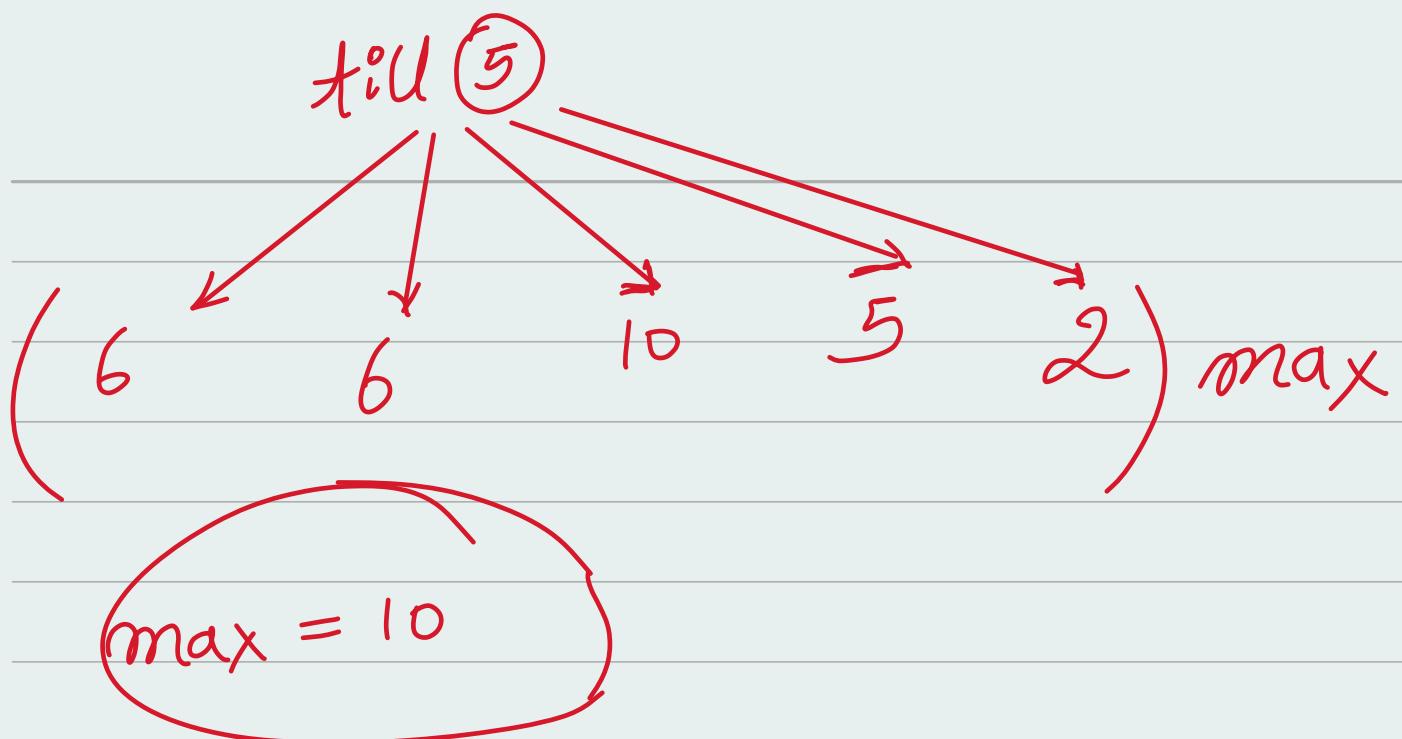
Suppose the index 1 had value 2 so

in that case we can move right but not in this case)



(we can move to right bar cause the value is 6 and the current value is 5. So total area under histogram is 10)

max = 10



Google, Facebook (Meta)

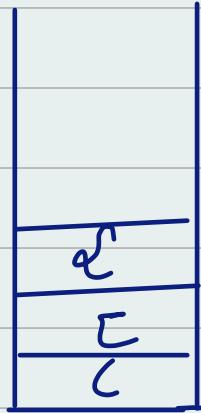
8) Determine if the input string is valid or not.

→ open brackets must be closed by the same type of brackets.

→ (), [], & { } → true

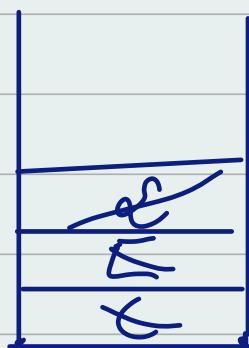
→ () → false

→ ([{ }]) = false



→ Check whether } is able to cancel the item at the top of the stack. In this case we have { so return false -

→ ([{ }])



The first closing } should be able to cancel the item at the top of the stack. In this case it is true - otherwise return false.

Code : Valid Parentheses?

class validParentheses {

public boolean isValid (String s) {

Stack < Character > stack = new
Stack <> ();

make a list of characters ([{ }]) and run a loop
for (char ch : s. toCharArray ()) {

if (ch == '(' || ch == '[' ||
ch == '{') {

stack.push (ch);

else {

if (ch == ')') {

if (stack.pop () != "("
|| stack.isEmpty ()) {
return false;

connection

make (stack.isEmpty ()
) ||
stack.pop () != opening

if (ch == ']') {

if (stack.pop () != "["
|| stack.isEmpty ()) {
return false;

```
        }  
    if (ch == ']') {  
        if (stack.pop() != '['  
            || stack.isEmpty())  
            return false;  
    }  
}
```

```
}  
return stack.isEmpty();
```

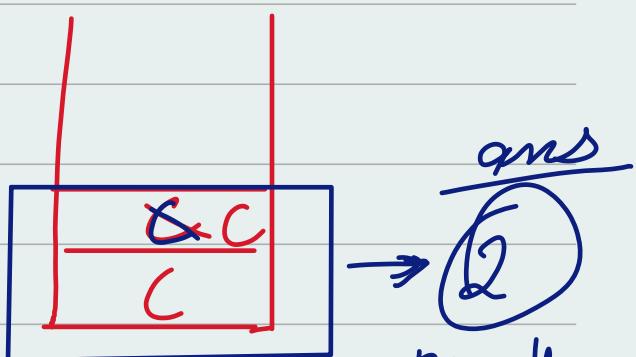
```
}  
}
```

8) Minimum Add to make Parenthesis

valid.

→ (() (

↑↑↑↑



parentheses
needed:

Code:

class Solution {

public int minAddToMakeValid
(String s) {

Stack < Character > stack = new
Stack <> ();

for (char ch: s.toCharArray()) {

if (ch == ')') {

if (!stack.isEmpty() &&
stack.peek() == '(') {

stack.pop();

} else {

stack.push(ch);

} else {

stack.push(ch);

} }

return stack.size();

y y

