PEPCODING

PURSUIT OF EXCELLENCE AND PEACE

MODULE

STACKS AND QUEUES

+91 11 4019 4461

PepCoding, 3rd Floor, 15 Vaishali, Pitampura Opposite Metro Pillar 347, Above Karur Vysya Bank, New Delhi, Delhi 110034

www.pepcoding.com

/pepcoding

Date
DELTA Pg No.

Stacks and Queues questions

Q. String compare after deletions. (Stack)

What?

Compare two strings s and t where strings contain lowercase and '#' as character; '#' deletes previous character. After deletion, compare the strings and return true if equal.

How?

Take two stacks, one for each string. Insert / Push char if char at i ≠ '#', else pop the top element.

ex: s: z b # c

t: z d # c

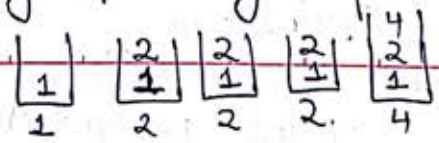
Now pop characters from both stacks together; if char not same, return false. If all same, return true.

st1	st2	
ch1 = c	ch2 = c	✓ pop 1
ch1 = z	ch2 = z	✓ pop 2.

// stack empty now, return true.

Q. # Variation: Remove digits that are repeated.

→ Instead of popping top during duplicacy, just don't insert new element.

ex: 1224 ⇒  = 124

Q. Print binary numbers. (Queue)

What?

Given a positive integer N, print all numbers from 1 to N in binary form.

How?

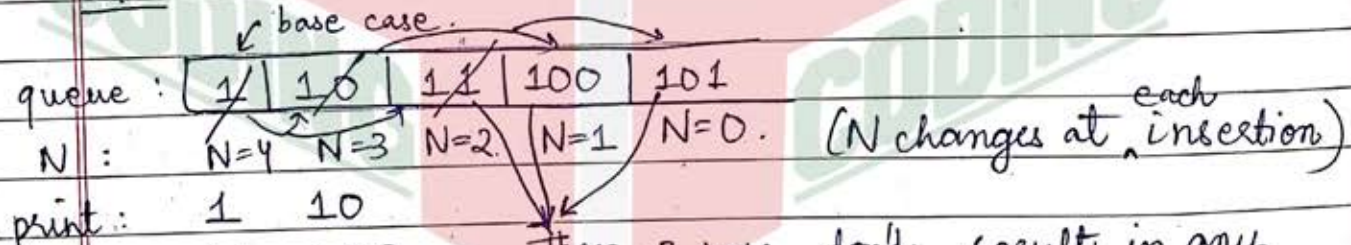
→ Use a queue and push '1' initially, and decrement N by 1 (∵ '1' is already 1 of the N answers.)

→ Now for one removeFirst() from queue, add a '0' and a '1' as 2 distinct numbers after the removed element's string and decrement n by 2 (∵ 2 numbers inserted.)

ex: If top was '10', add 2 numbers (as strings) '100' and '101' in queue.

// Do this if n != 0.

ex: N = 5.



print : 1 10 11 100 101.

These 3 pops don't result in any more additions to queue, so just print directly.

pseudo:

```
q.push("1"); n--;
while (n != 0)
    string rm = q.removeFirst();
    if (n != 0)
        q.addLast(rm + "0");
    if (n != 0)
        q.addLast(rm + "1");
    // print rm.
while (q.size() > 0)
    // print q.removeFirst();
```

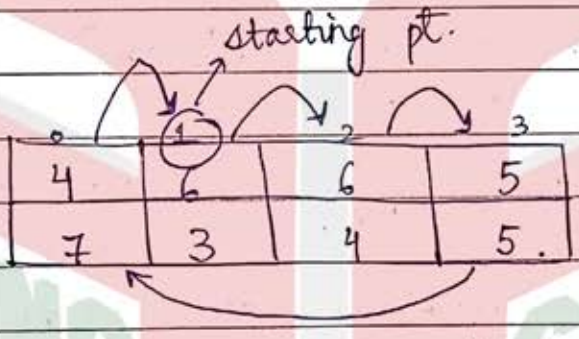
Q. Driving a Mustang.

What?

You have a Ford Mustang and there is a circular racing circuit in city where you want to test car. There are N petrol pums present on the circuit at various distances. You are given 2 arrays, one contains amount of petrol (in l.) at ith petrol pump and second has distance (in units) of ith petrol pump from (i+1)th petrol pump.
Mustang will stop at each petrol pump and have infinite capacity for storing petrol.
Return the position from where Mustang should start its journey in order to complete 1 round of track.
Mustang covers 1 unit of distance in 1 l. petrol.

How?

ex:



Start at index 0: petrol gained = 4 but you need to cover distance = 7.

$\Rightarrow$ petrol = −3.

Start at index 1 since index 0 gave -ve:

petrol = petrol gained − distance till next pump
= 6 − 3 = 3.

Continue at index 2: petrol = 3 + 6 − 4. = 5.

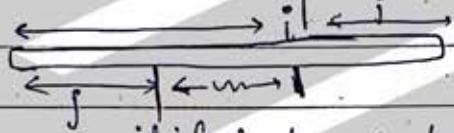
Continue at index 3: petrol = 5 + 5 − 5 = 5

Continue at index 0: petrol = 5 + 4 − 7 = 2.

(circular) Now you've reached index 1 again.

* Initially spare = 0.
* After each index, spare += petrol[i] - distance[j] if spare >= 0.
* Keep two pointers, i at starting pt, j for continuing till you can using that start pt.
* Increment j till either spare < 0 (if i=j, spare=0) and if j < i → break.
(-ve end case because i will have traversed the part where j is)

ex:



if j fails here, don't check again as area b/w j to i was already a fail.
* Return answer if spare ≥ 0 and j == i after an entire circular trip.

8. Playing the game. (Stack)

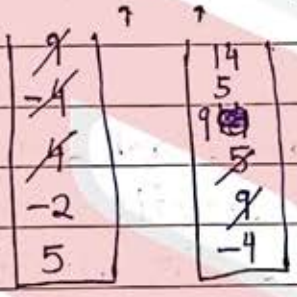
What?

You are playing a game where a string is given as:
'+': points scored in this round is sum of valid points in previous two rounds.
'D': points scored in this round is double the valid points scored in previous round.
'C': points scored in previous round is invalid and score must be removed.
Return total points scored after completion of all rounds.

How?

If char at a position is:
'C' : remove top element (pop)
'D' : push (st.top() * 2)
'+' : pop 1, pop 2 (pop 2 elements),
create sum = pop1 + pop2.
Now add back all 3 elements :
st.push(pop2); st.push(pop1); st.push(sum);
any other case: push the number in stack.

ex: 8
5 -2 4 C D 9 + +
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑



Answer = 14 + 5 + 9 + 5 - 4 - 2
= 27

Q. Adjacent - twins (stack)

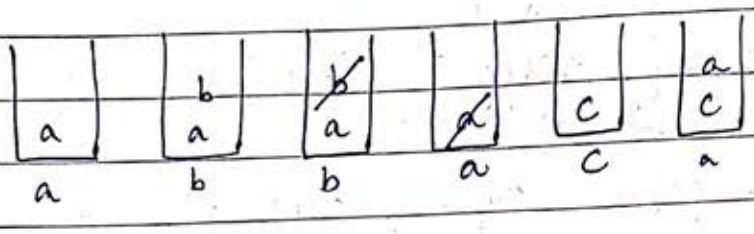
What?

Given a string S, duplicate removal consists of choosing two adjacent and equal letters, and removing them.
Make repeatedly removals as long as you can.
Return final string after all duplicate removals are done.

How?

Insert each character in stack and if the character to be inserted is same as top character, pop; else push it.

ex: s: "abbaca"



Ans = ca

Q. Consecutive pairs in an array. (Stack)

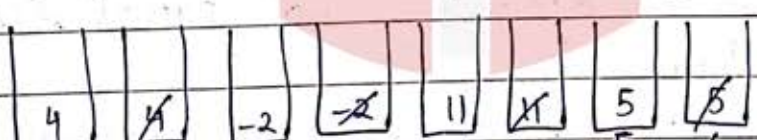
What?

Given a stack of integers N, check whether numbers in stack are pairwise consecutive or not. Pairs can be increasing / decreasing, and if stack size is odd, the element at top is left out.

How?

Just insert every even index in stack, and every odd index, check if it is consecutive with top element or not. If not, push it also in stack. In the last, if stack is not empty, return false, else true.

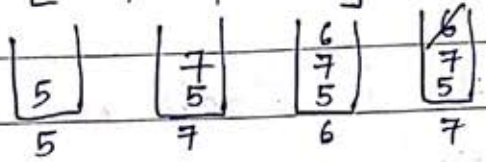
ex1: [4, 5, -2, -3, 11, 10, 5, 6, 20]



(20 is neglected)

↳ Stack is empty in last ⇒ true.

ex2: [5, 7, 6, 7]



⇒ Stack size == 2, return false.

Q. Fun with Asteroids (Stack).

What?

Given an array 'asteroids' of integers, representing asteroids in a row, for each asteroid, absolute value represents size and sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at same speed. Find state of asteroids after all collisions. If two asteroids meet, smaller one will explode. If both of same size, both will explode. Two asteroids in same direction never meet.

How?

ex: [5, -2, 7, -8, 4, -7, 2, 10]

Positions are honoured as:

5 → ← -2 : collide and answer = 5 till now. [5]

5 → 7 → : [7, 5]

5 → 7 → ← -8 : -8 collides first with 7, then 5 ⇒ [-8]

-8 ← 4 → : [4, -8]

-8 ← 4 → ← -7 : -7 collides with 4. [-7, -8]

-8 ← -7 ← 2 → : [2, -7, -8]

-8 ← -7 ← 2 → 10 → : [10, 2, -7, -8] : answer.

Q. Killing Process. (Queue)

What?

Given n processes, each has unique PID (process ID) and (parent process id) PPID. Each process has only one parent process, but may have more than one children processes like a tree structure. Process with PPID = 0 which means this process has no parent. You are given two lists as PID and PPID. You are also given a PID representing a process you want to kill, return a list of processes that will be killed in the end. Assume that if a process is killed, all its children processes are killed.

How?

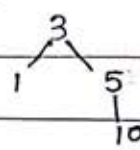
* Create a HashMap of <Integer, ArrayList<Integer>> where in front of each PPID are stored all its pid's (children process ids).
* Now use a BFS algorithm where initially you add the pid of the process to be killed in queue.
* Follow this algorithm now:

```
while (q.size() > 0)
    // removeFirst. (remove)
    // add it in List (work)
    // add all its children in queue using map. (add children)
```

* return list in end.

ex: PID: 1 [3] 10 5 (root node.) PPID: 3 0 5 3

HashMap: 3 → [1, 5]
5 → [10]



BFS: Id to be killed is 5.

1) q: ~~5~~ 2) ~~10~~ 3) No more to be added.

list: [5 10] ⇒ answer = [5, 10]

Q. Knight in Chessboard (Queue)

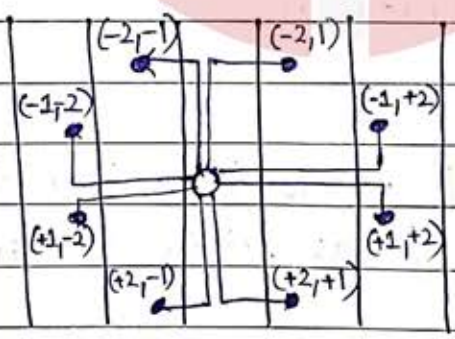
What?

Given a square chessboard of N x N size, the position of knight and position of target is given. Find minimum steps knight would take to reach the target position.

How?

* Use BFS algorithm whenever minimum moves / steps from a source to a target are needed.
* BFS algorithm works as:
  - Insert starting position of knight in q.
  - while (q.size() > 0)
    - → get the first from q and remove First. (q.)
    - → remove the first one. (r) → (C++)
    - → mark the box visited, if already visited, continue. (m*)
    - → if indexes of removed box are same as target, return the answer calculated. (w) → as level
    - → add all the valid boxes which can be reached from removed box. (a*) (Check all 8 directions)

Directions for a given box:



Maintain Pair Class as:

Pair
- int x
- int y
- int level.

* For source / start, level is 0, and when a child is added in a*, it is added as level = level of removed box + 1.

Q. Rotten crops. (Queue)

What?

Given a matrix of dimension r X c, it can have values as 0, 1, 2 where

0 : empty cell

1 : cells with fresh oranges.

2 : cells have rotten oranges.

Return minimum time to rot all oranges.

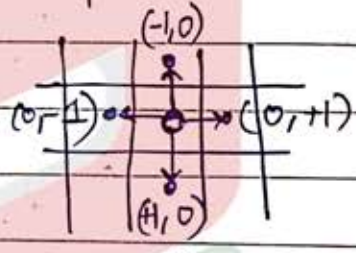
A rotten orange can rot other fresh oranges in up, down, right and left directions in unit time.

If it is impossible to rot every orange then return -1

How?

*Use BFS algorithm as you need to find minimum time.

* Algorithm goes as:
  - get
  - remove
  - mark rotten
  - add neighbours which are valid (4)



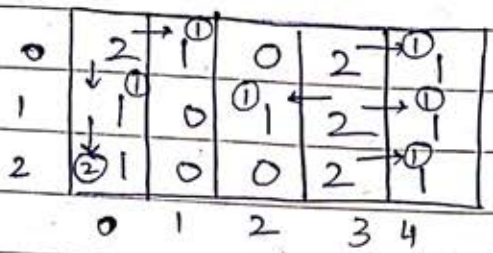
* Initially add all those boxes in queue which are marked 2 (that is they are rotten :)

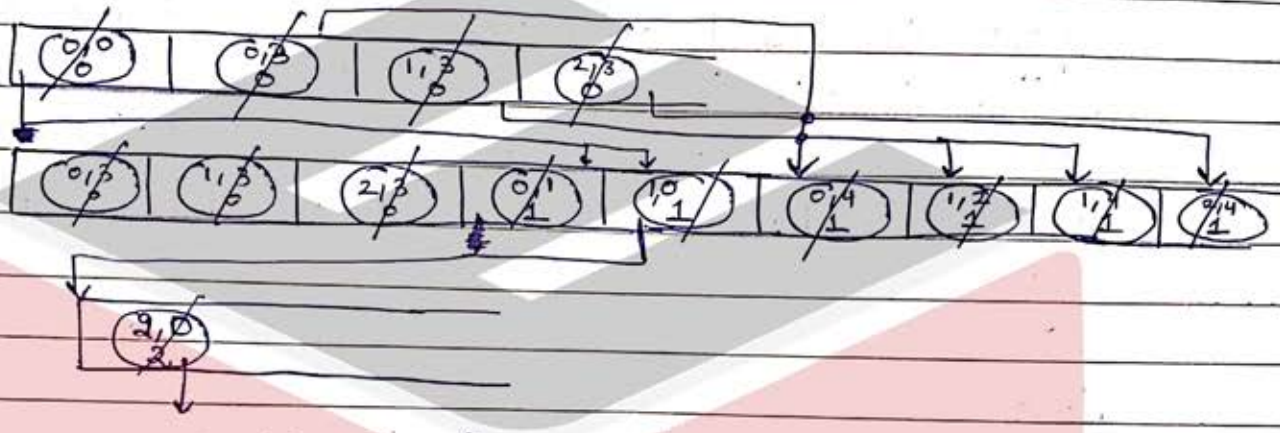
* Class is implemented as:
  - ↳ int x
  - ↳ int y
  - ↳ int time // which tells that this (x,y) box // was rotten at time = time.

* After BFS is completed, check if their are any more fresh oranges in the matrix, if yes, return -1, else the time calculated.

ex: 3, 5

2 1 0 2 1 1 0 1 2 1 1 0 0 2 1





answer = 2.

Q. Sort the Stack (Stack)

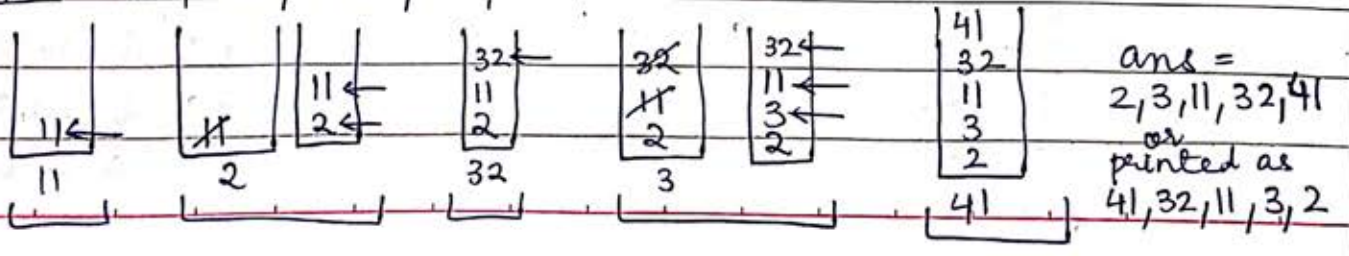
What?

Given a stack, task is to sort the stack such that top of stack is greatest element.

How?

If current element > st.top(), push it, else while it is not at its right position, pop elements, insert current element and then push back all the popped elements in same order as popped.

ex: 11, 2, 32, 3, 41:



Date
DELTA Pg No.

Q. Function Time (Stack)

What?

You are given various logs as timestamps of a function thread as 0: start :3 which means function with id 0 started at beginning of timestamp 3, and 1: end: 2 means function with id 1 ended at end of timestamp 2. Function's exclusive time is number of units of time spent in this function excluding recursive calls to child functions.

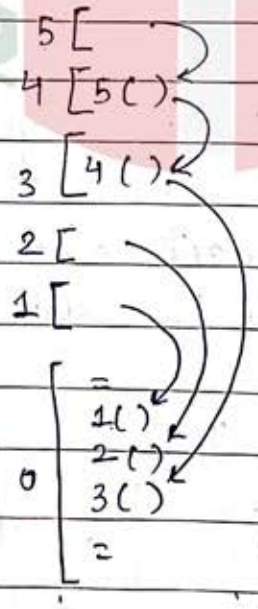
CPU is single threaded; i.e., one function executed at one time. Return exclusive time of each function sorted by id.

How?

* If process starts, push in stack.
* If process ends, its time is end time − start time +1 − not my time (nmt) where not my time is the time of child functions in it. After pop, add back nmt as nmt += end time − start time + 1.

ex:

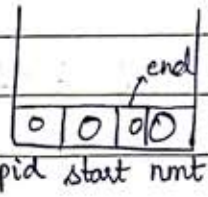
1) 0 : s : 0
2) 1 : s : 2
3) 1 : e : 5
4) 2 : s : 6
5) 2 : e : 9
6) 3 : s : 10
7) 4 : s : 14
8) 5 : s : 16
9) 5 : e : 18
10) 4 : e : 20
11) 3 : e : 23
12) 0 : e : 30



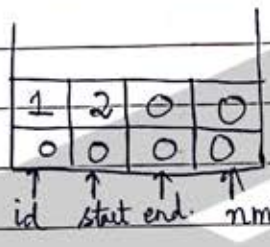
Keep pair in stack having ~~int~~ implementation as:

Pair
- int id
- int start
- int nmt // not my time
- int end

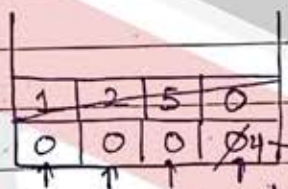
1) 0 : s : 0



2) 1 : s : 2.

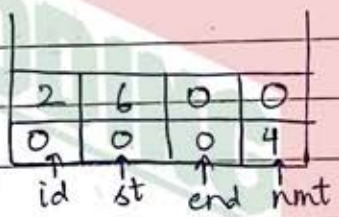


3) 1 : e : 5.

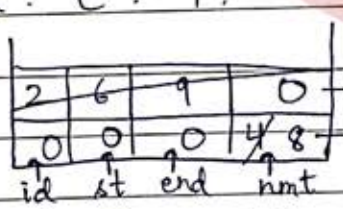


→ Popped, 1's time = 5 - 2 + 1 = 4.
→ 0's nmt += 4 because it was time of its child function 1( ).

4) 2 : s : 6.

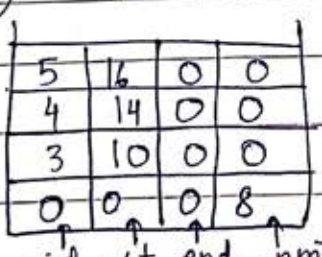


5) 2 : e : 9.



→ 2's time = 9 - 6 + 1 = 4
→ 0's nmt += 4 ⇒ nmt = 8.

6), 7), 8) 3 : s : 10 , 4 : s : 14, 5 : s : 16



9) 5 : e : 18

5	16	18	0
4	14	0	3
3	10	0	0
0	0	0	8

⇒ 5's time = 18 − 16 + 1 = 3

⇒ 4's nmt += 3 .

10) 4 : e : 20

4	14	20	3
3	10	0	7
0	0	0	8

⇒ 4's time = 20 − 14 + 1 − 3 = 5

⇒ 3's nmt += 7 (20 − 14 + 1)

11) 3 : e : 23

3	10	23	7
0	0	0	8
			22

⇒ 3's time = 23 − 10 + 1 − 7 = 7

⇒ 0's nmt += 14 (23 − 10 + 1)

⇒ nmt = 22

12) 0 : e : 30

0	0	30	22

⇒ 0's time = 30 − 0 + 1 − 22

= 9

∴ Answer :

Id → Time

0 → 9

1 → 4

2 → 4

3 → 7

4 → 5

5 → 3

8 First Negative Value in a K sized Window

What?
Given an array of Integers and a value K. Print the first negative integer of each and every window of size K. If no negative number in window then print 0.

How?
We will keep 2 indexes i & j
j will point to the negative no. and
i represent start window.

a) Move j till you get a negative no.
b) Check if j (idx) lies within the range of current window starting from i ie if j <= i+K-1 then res[i] = arr[j]
c) If by chance j < i move j = i and repeat above steps.

K = 5

eg:

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	-1	-2	4	-8	5	6	7	8	-10	3

Ans: -1 -1 -1 -1 -2 -8 -8 -10 -10
(3 <= 0+5-1)
↳ last window

For i = 0 to 3 j remains at 3 and res is -1
For i = 4 j was < i so j = i and now check.
For i = 5 to 6 j remains at 6 and res = -8
For i = 7 to 8 j remains at 12 and res = -10

Date
DELTA Pg No.

Q Decode a string at Index

What?
You are given an encoded string and an index K. The given string is read character wise. and is being written on tape.
If character is letter than it is written as it is.
If the character is digit then the whole string upto its current form is written again d-1 times.
Return the Kth character of decoded string.

How?
1) Calculate the resultant decoded string length
Initially size=0, when character comes just increment size else in case of digit size * digit.

2) Traverse in the encoded string from last
If we are shortening the decoded string virtually by discarding some parts of the string where answer won't be.

```
i=s.length()-1 to 0.
    K=K%size  //shorten string and adjusting K accordingly
    if(K==0)
        return ch;
    else
        if(ch>='0' && ch<='9')
            size /= digit  //(dividing the str's length and shortening it)
        else
            size --  //in case of char.
```

Date
DELTA Pg No.

i (0) (1) (2) (3) (4) (5)

eg: a b 2 c 2 d 3 x y.

K=4 K=9 K=9
size=4 size=5 size=11

① size = ~~35~~ ~~35~~ ~~35~~ 11 10 ~~5~~ ~~4~~

② K = ~~20~~ 9 4
20 % 11 9 % 5

at i = 2
K % size = 4 % 4 = 0
here char(P) = 'b'
is Ans

same string was repeated

ab abc ababcd ababcababcd ababcababc
d x y.

Ans = b

Q Unique characters (Queue)

What?
Given an input stream of N characters consisting only lowercase letters. The task is to find first non repeating character each time a character is inserted to stream.

How?
Maintain a frequency map and a Queue. Queue contains first unique character

eg: a b a c c d b e d f e.
0 1 2 3 4 5 6 7 8 9 10

1) Frequency update 2) Dequeue

Queue [a|b|c|d|e|f]
(at 2) (at 6) (at 6) (at 8) at (10)

a → ✓2
b → ✗2
c → ✗2.
d → ✓2
e → ✗2
f → 1

Ans a a b b b b d d e f

1) Frequency update
2) Dequeue from Queue till first character (front char) has frequency = 1 or Queue is not empty.
3) Enqueue only when frequency is 1.

By using queue first non repeating character is at front.

Rotation of Cards (Queue)

What?

Given a sorted deck of cards 1 to N. We pick first card and put it at the back of deck. Now we pick another card it is first card, we put it outside of deck and so on. Now pick 2 cards and put at back and pick card. We perform these operation successively till deck is empty. Give which card is first, second ... picked.

How?

We use queue to perform given activity.

eg ⇒ n=5 (Intial push 1 to n in queue)

element vs position

(I) 1 → 2 → 3 → 4 → 5 → 1 — 2 → 1

(II) 3 → 4 → 5 → 1 → 3 → 4 — 5 → 2.

(III) 1 → 3 → 4 → 1 → 3 → 4 — 1 → 3, 3 → 4.
① ② ③ ④.

(IV) 3 → 4 — (rotations = 4 %. 2 = 0 so 0 rotations) — 4 → 5

(V) 4 — rotations = 5 %. 2 = 0 so 0 rotation

eg n=6

(I) 1 → 2 → 3 → 4 → 5 → 6 → 1 — 2 → 1

(II) 3 → 4 → 5 → 6 → 1 → 3 → 4 — 5 → 2.

(III) 5 → 6 → 1 → 3 → 4 — 4 → 3

(IV) 6 → 1 → 3 → 6 — 1 → 4

(V) 3 → 6 — 6 → 5

(VI) 3 — 3 → 6

Q Fleet of Cars

What?

There are N cars going to the same destination along a lane road. The destination is target km away. Each car has a constant speed (in kmph) and initial position in km towards the target along the road. A car can never pass another car ahead of it but it can catch up to it i.e they will have same position now onward and drive it with same speed) A car fleet is some non empty set of cars driving at same position and same speed. ~~If they..~~

How?

Make a pair class that stores ~~time~~ (speed / distance) speed & distance. (time is time needed to reach destination)

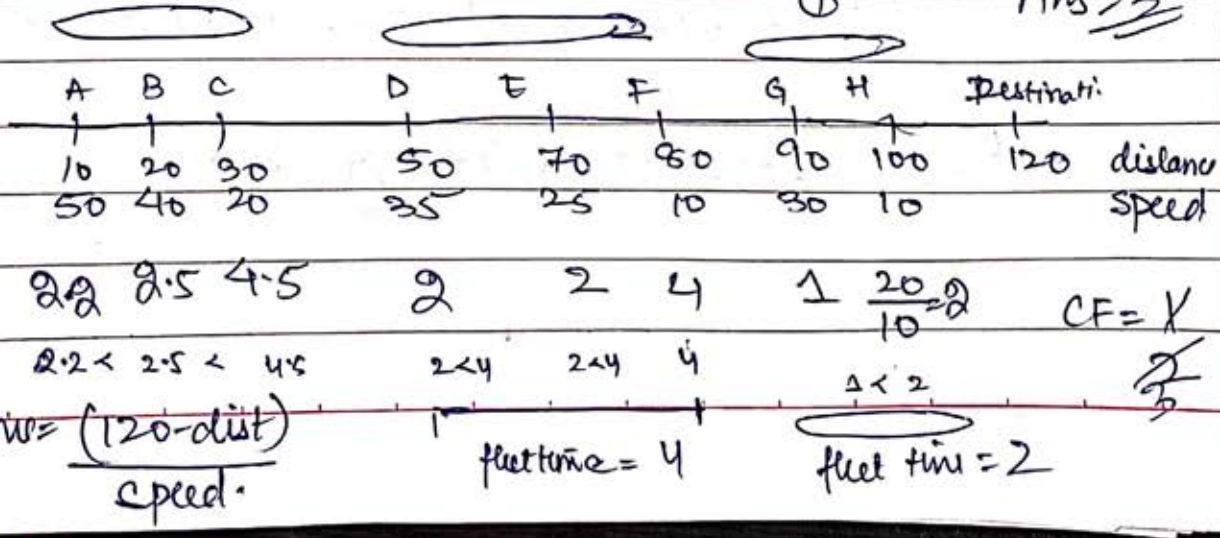
Sort the Array of such pairs on the basis of distance.

~~From left~~ right to left If i-1 time is greater than maximum time of that fleet. than it will form a new fleet i.e fleet count is incremented.

If i-1 time is less than equal to 'time of fleet time than it will be part of earlier fleet.

Ans = 3

eg:



Q K Reverse in a Queue. (Queue)

What?
Given an Integer K and queue of Integers, we need to reverse the order of the first K elements of the queue leaving the other elements in the same relative order.

How?
1) Remove K elements from queue and push in stack
2) Now pop from stack and add (enqueue in queue) till stack is empty.
3) Now dequeue from queue & enqueue in queue n-K elements.

eg: [a b c d e f g h i j] Queue.
K=4

① [d c b a] (dequeue from queue & push in stack)

② e→f→g→h→i→j→d→c→b→a
(Pop from queue & enqueue)

③ ~~e→f→g→h→i~~→d→c→b→a→e→f→g→h→i

Ans d→c→b→a→e→f→g→h→i

Date
DELTA Pg No.

8 Celebrity Problem (stack)

What?

You are at a party of N people where there may be a celebrity know to everyone but doesn't know anyone. Find the celebrity in party.

How?

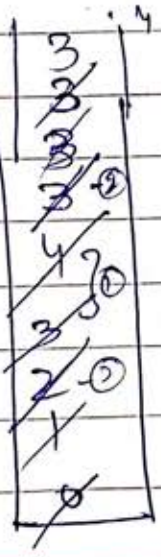
Add 2 to N (0 to N-1) in stack. Pop 2 element from stack val1 = first popped person, val2 is second popped person.

If val2 knows val1 (check the grid) push val1 else push val2.

For last remaining element in stack check grid to confirm that it knows no one & known by every one. This is done because while checking in stack only certain (*) marked boxes where checked not all.

eg:

	0	1	2	3	4
0	X	T	T	T*	T
1	F	X	T	T*	T
2	T	T	X	T*	F
3	F	F	F	X	F*
4	T	T	T	T*	X



→ Here if True was present then no celebrity.

3 doesn't know 4 so 4 not celebrity.

2 know 3

1 know 3

0 knows 3.

Checking for 3.

Ans celebrity is 3

Date
DELTA Pg No.

Q Remove K to make Smallest (Stack)

What?

Given a non negative integer num represented by a string. Remove K digits from the number so that new number is the smallest possible. (order need to be maintained)

How?

eg: 1 4 6 2 1 2 5 4 9 K=3

(we will keep smallest digit to most significant pair (to left))

9
4
5
2
1
~~2~~
~~6~~
~~4~~
1

* when no./digit < top & current pop < K pop from stack and increment current pop.

1 pops 2 as 2>1 and current pop < K current pop = 3.

2 is less than 6 so pop same with 4 current pop = 1 → current pop = 2.

Ans = 112549

We are keeping smallest digit at msb because it will make no. of smaller.

@@ eg: 4 6 7 3 7 8 9 K=5

9	
8	
7	
3	
~~7~~	K=4
~~8~~	K=3
~~9~~	K=2

since K is left we will now pop from top of stack. Because target will be less that way.

Ans = 37 (89 > 37)

Date
DELTA Pg No.

8 Reverse a Queue (Queue)

What?
Reverse the contents of Queue.

How?
1) Dequeue from Queue and push in stack all elements
2) Now pop and enqueue in queue.
3) Queue is reversed.

eg:

9 Ternary operator (Stack)

What?
Given a ternary expression solve it and print the result.

How
1) Any numbers → Push in stack
2) colon → ignore.
3) ? → ~~Ignore~~ Push in stack
4) T/F → Pop 2 elements from stack if 'T' then (If ? on top) push first popped else push second popped.
5) T/F when question not on top push.

(solve Right to left

eg: T ? F ? T ? 1 : 2 : F ? 3 : 4 : T ? 6 : 7 ←

Q Stack Validation (Stack)

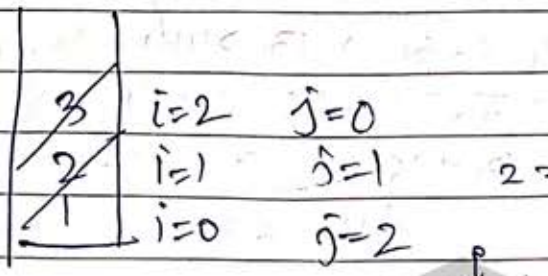
What?
Given 2 sequence pushed & popped with distinct values, return the true if and only if this could have been the result of sequence of push & pop operations on an initially empty stack.

How?
1) Push From the push sequence & increment it index.
2) If top of stack equal to pop current element pop till current element of pop equal to top of stack.
3) ~~Else~~

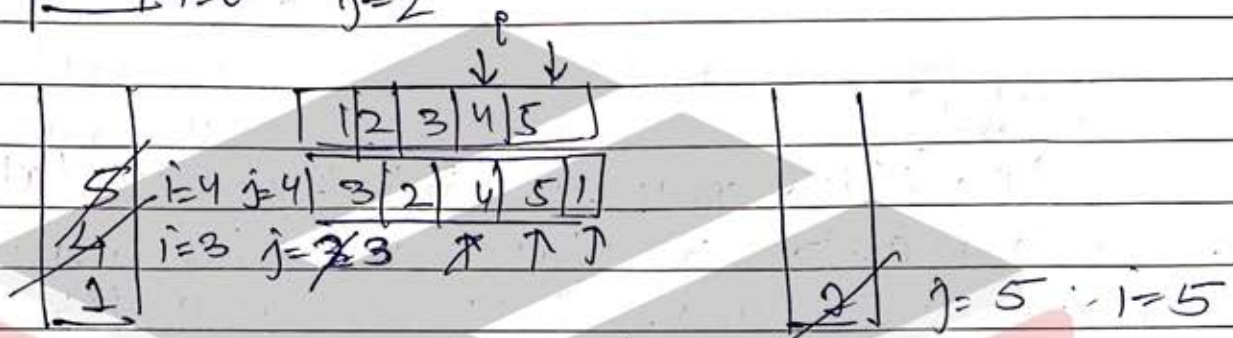
eg: Push [1, 2, 3, 4, 5] (T)
Pop [3, 2, 4, 5, 1]

3	i=2 j=0 top = arr[j]
2	i=1 j=0 top != arr[j]
1	i=0 j=0 ~~(pop!=)~~ top != arr[j]

(II) Pop till pop == pop[j]



(III)



At last stack empty & i & j both equal to length so possible sequence.

Q Connecting Ropes

What?

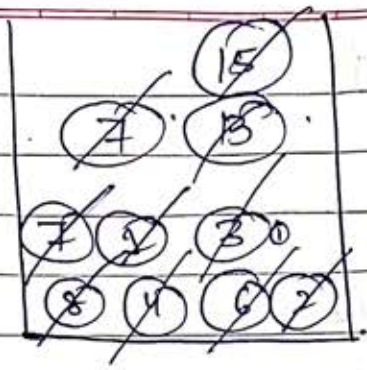
There are given N ropes of different lengths, we need to connect these ropes. The cost to connect two ropes is equal to sum of their lengths. Find the minimum cost to connect these ropes.

How

Make a priority queue (Min Priority Queue) and add all pieces to it.

```
while (pq.size() != 1)
{
    long l1 = pq.remove()      (similar to huffman
    long l2 = pq.remove()       encoding)
    sum += l1+l2
    pq.add(sum)
}
return sum.
```

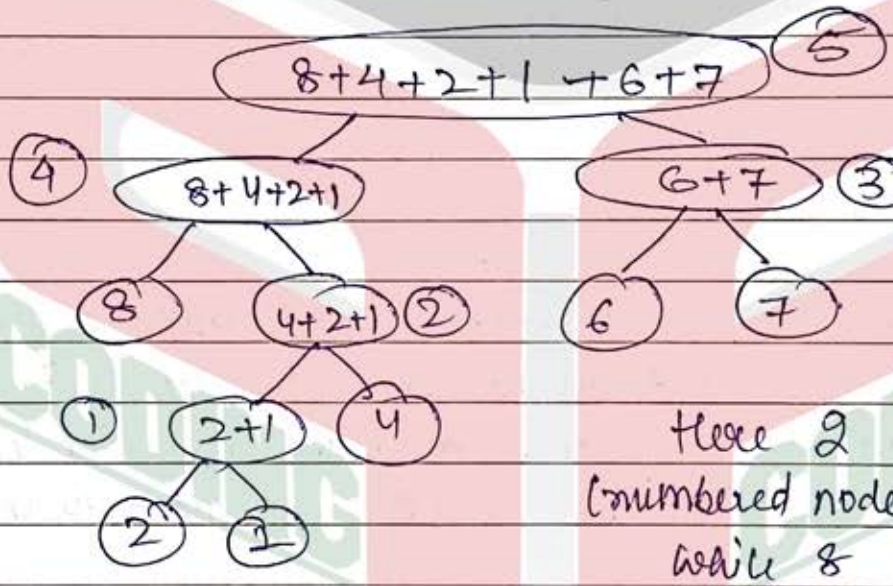
eg: 8 4 6 2 7 1



① 1+2 = 3 push
② 3,4 ⇒ 7 push
③ 7,6 ⇒ 13 push
④ 7,8 → 15 push
⑤ 13+15 = 28 cost

Ans = 66

Why?
No. which come in Priority Queue late or is greater is added least time hence resulting in less sum. So we use min pq. and multiple times add smaller no.



Here 2 occured in (numbered nodes) 4 times while 8 occured 2 times only

⑧ CPU tasks Schedule

What?
Given an array representing tasks CPU needs to do. Letters are different tasks. Tasks could be done without original order. Each task could be done in one interval For an interval CPU can be idle or can complete a task.

There is a non negative cooling interval in that means 2 same tasks can't be done together. Find the least intervals the CPU will take to finish the given tasks.

How?

1) Make a frequency map of the tasks.
2) Make Pq (Priority Queue) that stores pair (Task, freq)
3) While Priority Queue (Max Pq) size is greater than zero, pop K+1 elements from pq and push with updated freq (if zero then don't push)

eg:

A A A A A B B B B B C C C D D E

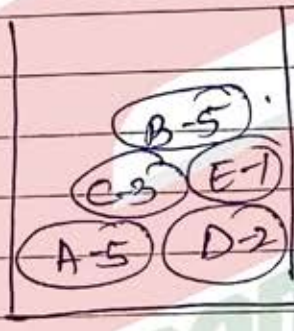
K=3.

FMAP

A→5 D→2

B→5 E→1

C→3



Ans

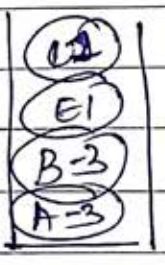
(I)

A	B	C	D
0	1	2	3
A5	B5	C3	D2

count = 4

(II)

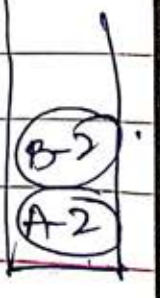
A	B	C	D	A	B	C	D
0	1	2	3	4	5	6	7
A5	B5	C3	D2	A4	B4	C2	D1



count = 8.

(III)

A	B	C	D	A	B	C	D	A	B	C	E
0	1	2	3	4	5	6	7	8	9	10	11
A5	B5	C3	D2	A4	B4	C2	D1	A3	B3	E1	



Count = 12.

Count += .

Date / DELTA Pg No.

(IV)

A	B	C	D	A	B	C	D	A	B	C	E	A	B	X	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

count = 16 (when elements are less than x+1 then x+1 cells are used)

B-1
A-1

(V)

A	B	CD	AB	CD	AB	CE	AB	X	X	AB	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

count = 18

Date
DELTA Pg No.

Q Largest Min Pair (Stacks)

What?

Given an array of Integers find the sum of the smallest elements & second smallest element for all subarray and return maximum sum.

How?

We will consider window of 2 size.

eg:

4	5	2	7	6
0	1	2	3	4

All possible pairs. (subarray)

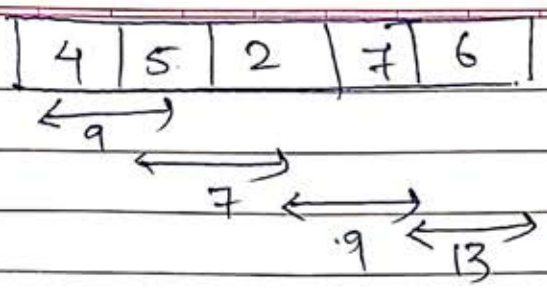
× 4 (not pair)	× 5 → (not pair)	2 → (not pair)	× 7 (not pair)	× 6 not pair
✓ 4 5 → 9 (sum)	✓ 5 2 → 7	✓ 2 7 → 9	✓ 7 6 → 13	
× 4 5 2 → 6	× 5 2 7 → 7	2 7 6 → 8		
× 4 5 2 7 → 6	× 5 2 7 6 → 7			
× 4 5 2 7 6 → 6				

Here we will consider window of 2 size only. because when an element is added in subarray it can be greater than smaller & second smallest in this case we want smallest & second smallest so no change in sum.

eg: 4 5 2 7 sum before addition of 7 was 6

After 7 addition second smallest & smallest remain same so no chang

eg: 4 5 2 before adding 2 (smaller no.) sum was 9. now it is 6 we need max sum Hence ignored.



Ans = 13
only (4,5) (5,2)
(2,7) & (7,6)
subarray are considered.

Next GREATER Element (NGE) BASED QUESTIONS

Q Next Greater Element (on right)

What?

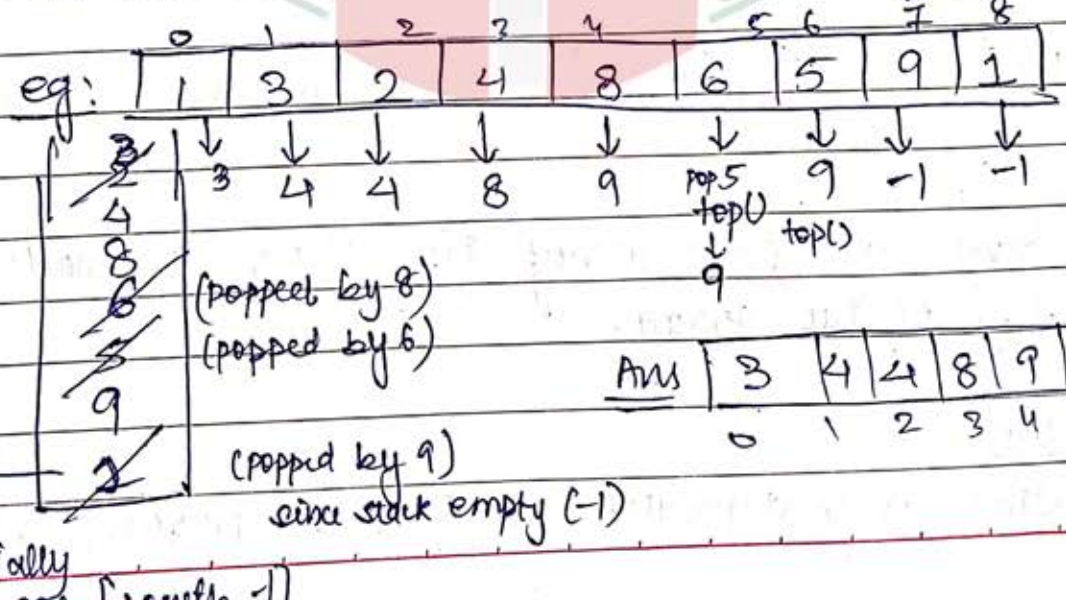
Given an array of Integers. Find the next greater elemenent for each element of array in order of their appearenca in array.

How?

1) Traverse array from right to left
2) While stack is not Empty and stack.top is less than equal to current element pop from stack
3) If stack empty no next greater element
4) Else top of stack is the next greater element for current element. 5) Push current on stack

Stack stores the potential greater elements for current element.

By popping those elements that are less than equal to current element. we excluding all those elements that can't be next greater.



Date
DELTA Pg No.

Q **Next Greater element on left**

What?

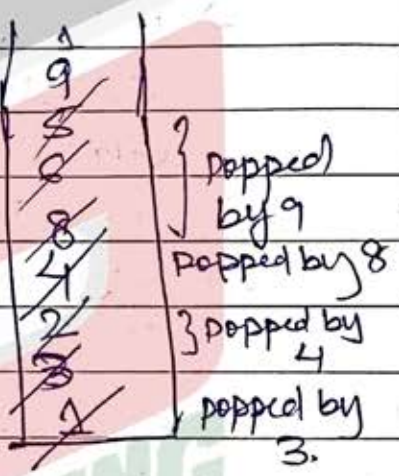
find next greater element for each element on left.

How?

Same as nextgreater element on right Just traverse from left. →

eg:

0	1	2	3	4	5	6	7	8
1	3	2	4	8	6	5	9	1
-1	-1	3	-1	-1	8	6	-1	9



* pop till arr[i] > top of stack ~~or st.size > 0~~
* If (st.size == 0)
  nL[i] = -1
  else
  nL[~~i~~] = top

Ans

-1	-1	3	-1	-1	8	6	-1	9

Q **Next Smaller on left**

What?

Given an integer array Find the next smaller on left of the array.

How?

Same as next greater on left Just popping condition

is changed. (Traversal from left to right)

eg:

0	1	2	3	4	5	6	7	8
1	3	2	4	8	6	5	9	1
-1	1	1	2	4	4	4	4	-1

~~6~~	popped by 5
~~8~~	popped by 6
~~4~~	popped by 1
~~2~~	popped by 2
~~8~~	popped by 1
~~1~~	

* pop till element of top is not less than current element and > size is 0

* If st.size() == 0
  ns[i] = -1
  else
  ns[i] = top.

Ans ns

0	1	2	3	4	5	6	7	8
-1	1	1	2	4	4	4	4	-1

Q Next Smaller on right

What?

Given an integer array. For each element find the next smaller element on right.

How?

Same strategy as next smaller on left Just traverse from ~~left~~ right to left.

eg:

0	1	2	3	4	5	6	7	8
1	3	2	4	8	6	5	9	1
-1	2	1	1	6	5	1	1	-1

~~1~~	
~~3~~	
2	
~~8~~	popped by 4
~~6~~	
~~5~~	
9	popped by 5
~~1~~	popped by

Same algo as above:

Ans = ns

0	1	2	3	4	5	6	7	8
-1	2	1	1	6	5	1	1	-1

Q Next Greater In other

What?
You are given 2 arrays (without duplicates) nums1 & nums2 where nums1 element are subset of nums2. Find all the next greater for nums1 elements in corresponding places of nums2.

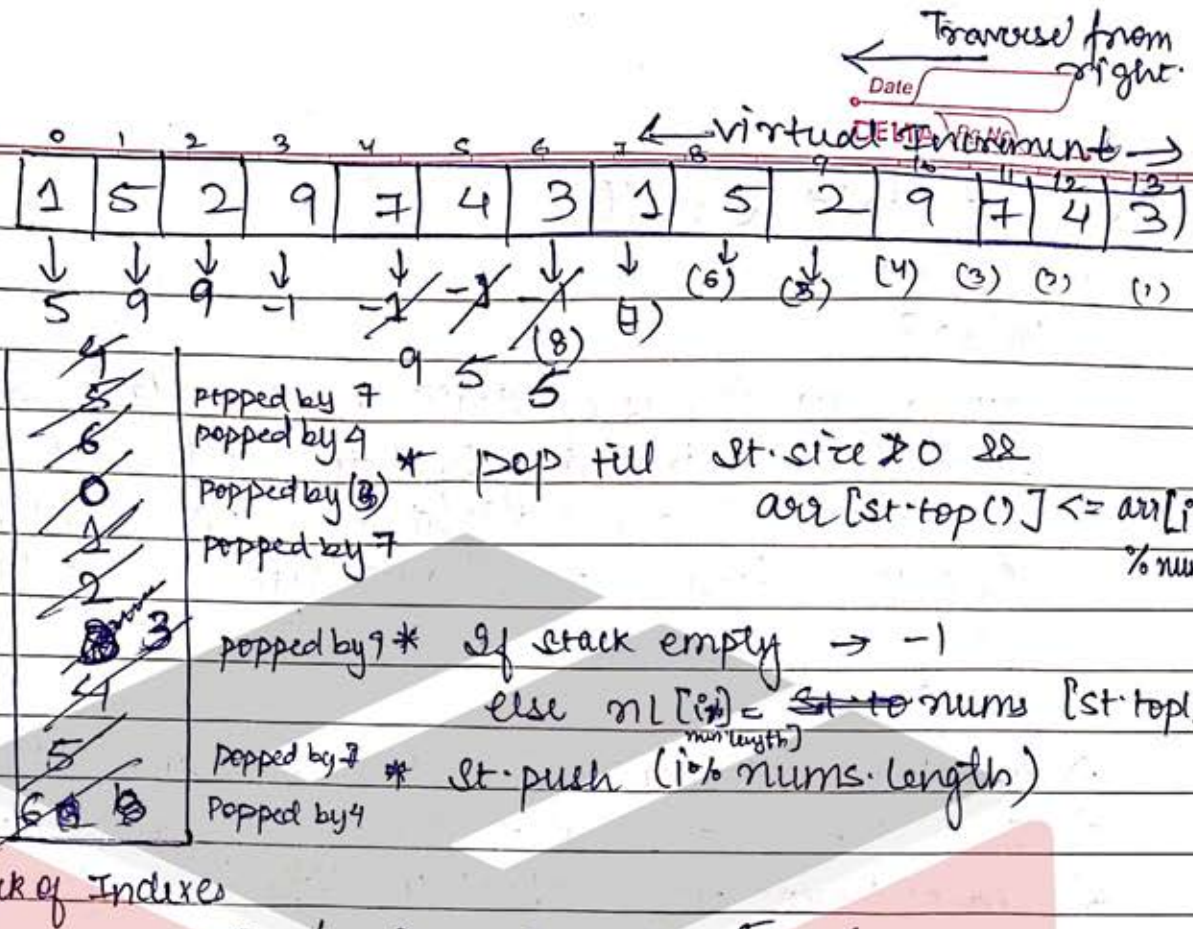
How?
1) Find NGE for nums2 array.
2) Store all elements as Key in HashMap of nums1.
3) While finding NGE if element ~~arr~~ nums1[i] exist in map then update the value for it. in hashmap.
4) Make a result array of next greater from hash map.

Q Next Greater in Circular

What?
Given a circular array the next element of the last element is the first element of array Find the next greater no. of every element and return array.

How?
eg: | 2 | 5 | 2 | 9 | 7 | 4 | 3 |

* consider this array as 2 repetition (2) of current array.
* Stack of indexes is made.



Q Next Warmer Day

What?

You are given an array having temperatures on various day, for each day you need to find out the number of days you have to wait for next warmer day.

How?

Just find the next greater Element for each day.

Make stack of indexes and res is the gap b/w next greater and current.

* Pop till arr[i] >= arr[st.top]
* res[i] = st.size() == 0 ? 0 : st.top() - i;

Date
DELTA Pg No.

Q **left Right smaller**

what?

Given an array of Integers the task is to find the maximum absolute difference b/w the nearest left & Right smallest element of every element in array. when not found then nS = 0

How?

1) Find the next smaller on left and store in array
2) Find the next smaller on right and store
3) calculate the absolute difference for each and find max.

eg)

	1	3	2	4	8	6	5	9	1
nsl	0	1	1	2	4	4	4	4	0
nsr	0	2	2	2	6	5	2	1	0
diff	0	1	0	1	2	1	3	3	0

(absolute diff)

max = 0 1 3

Ans = 3

Q ~~Find negative W~~

Q **Hight of the closest tower**

what?

Given n towers followed by their heights. For every tower find the largest closest tower towards right greater then height of tower. Print sum of such heights. No tower so height 0.

How?

1) Find the next larger Element for each tower.
2) Sum all the results for each tower.
3) Return result

Q Stock Span

What?

Find the span of the stock price for the current day. The span of stock's price today is defined as the maximum number of consecutive days (starting from today going backwards) for which the price of stock was less than or equal to today's price.

How?

Find next ~~smaller~~ larger on left (stack of indexes)
Span is idx - idx of n larger on left.

eg:

0	1	2	3	4	5	6	7	8	9
30	35	40	60	50	45	48	49	70	20
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1	2	3	4	1	1	2	3	9	1

(i - top)

9
8
7 popped by 70
6 popped by 49
5 popped by 48
4 popped by 70
3 popped by 70
2 popped by 60
1 popped by 40
0 popped by 35

* Pop till nums[i] >= nums[st.top()]
* If stack empty
  res[i] = i+1
* else
  res[i] = i - st.top()

Date
DELTA Pg No.

8 Maximum Rectangle Area Histogram

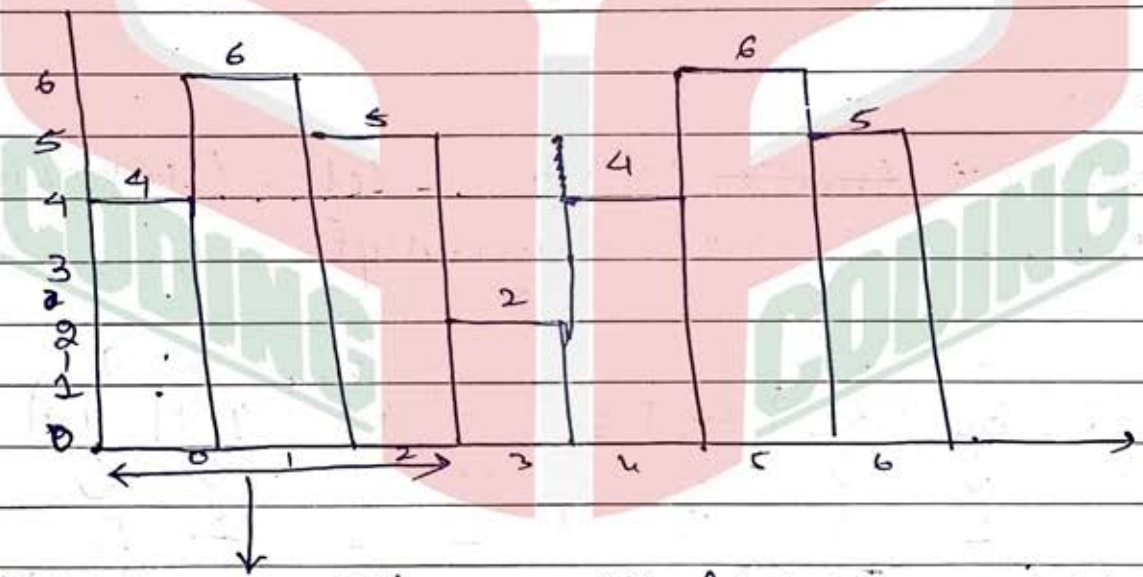
What?

Given an array of Integers where each value represent the height of bar. Assume the width of each bar as 1 unit. Find the maximum rectangle area in histogram

How?

We use technique of next smaller on left and next smaller on right. to find the area.

For eg: | 4 | 6 | 5 | 2 | 4 | 6 | 5 |



For area of 4th we will find ns on left which is nil and next smaller on right which is 2 all the area b/w both smallers is the area of rectangle with height = 4.
Here area is 12. $4 \times (3-0) = 12$.

To calculate above without finding ns. on left and ns. of right seperately we follow following algorithm.

→ stack of Integers (to store Indexes)
→ Calculation of area of particular height at the time of pop.

```
For i=1 to length(n)
    while (st.size() > 0 && ht[st.peek()] > ht[i])
        currentheight = st.pop()
        leftboundary = st.size()>0 ? st.top() : -1;
        curr area = (i - lb - 1) * height[curr height]
        max area = max(max area, cur area)
        //calculation at pop.
    st.push(i)

while (st.size() > 0)
    int curht = st.pop()
    int rightBoundary = ht.length
    int leftBoundary = st.size>0 ? st.top() ? -1
    curarea = (ub - lb - 1) * ht[curht]
    area = max(curarea, area)
}
```

5(6)
6(5) → popped by 5 (ub = 6, lb = 4) area = (6-4-1)*6 = 6 max = 0 6 10 12
4(4)
2(3)
5(2) → popped by 2 (ub = 3, lb = 0, area = (3-0-1)*5 = 10
6(1) → popped by 5 (ub = 2, lb = 0
4(0) cur area = (2-0-1)*6 = 6
val(idx)

popped by 2 (ub = 3, lb = -1, area = (3*4) = 12

max = 12 14

5(6)	→ ub = 7, lb = 4, area = (7-4-1)*5 = 10
4(4)	→ ub = 7, lb = 3 area = (7-3-1)*4 = 12
2(3)	→ ub = 7, lb = 7-1 area = (7-1-1)*2 = 14

Ans = 14

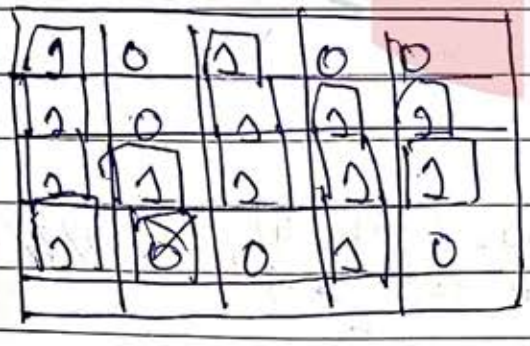
Q Rectangle in a matrix

What?

Given a matrix of '0' and '1'. Return the area of the largest rectangle formed by 1's in the matrix.

How?

1) Use max area histogram to find largest rectangle.
2) For each row apply max area histogram
3) Each row value will be changed if arr[i][j] = 1 then arr[i][j] += arr[i-1][j]. Now for this new row calculate max area histogram



⇒

1	0	1	0	0	→ 1
2	0	2	1	1	→ 3
3	1	3	2	2	→ 6
4	0	0	3	0	→ 4

6 6

maxarea = max(maxrectarea(row), maxArea)

Ans = 6

Date
DELTA Pg No.

Brackets Based Questions

Q Long Parenthesis.

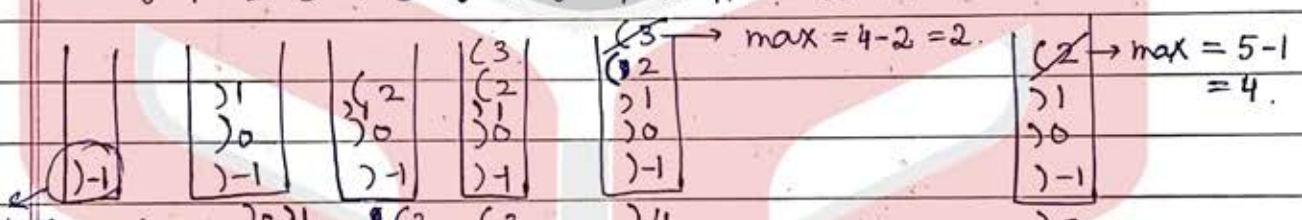
What?

Given a string containing opening and closing parenthesis '(' and ')', find length of the longest valid parenthesis substring.

How?

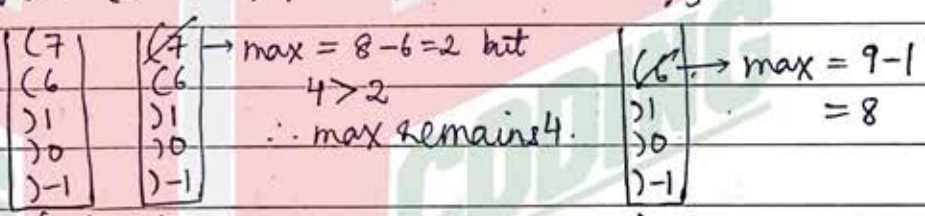
If '(' → always push in stack.
If ')' → settle and then push.

ex:)) (( )) (( )) ( ) (( ) (
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15.

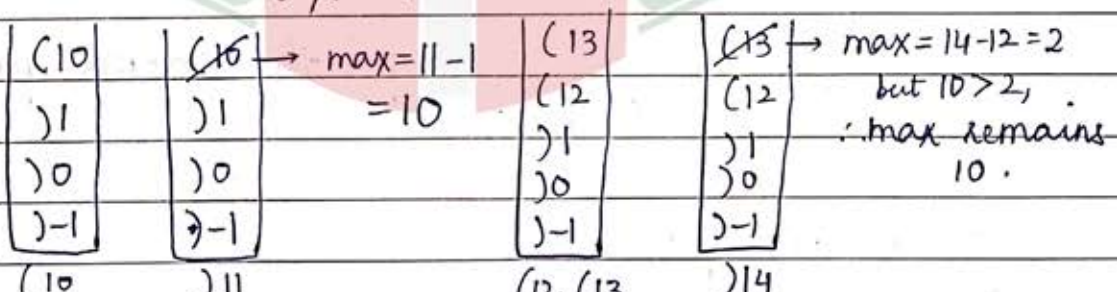


Initially push a closing, considering it was at -1 index.

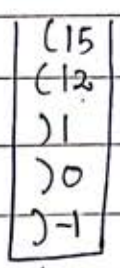
)0,)1 (2 (3)4)5



(6, (7)8)9



(10)11 (12, (13)14



(15

∴ In the end, we found maximum length = 10.

Q. Duplicate Brackets.

What?

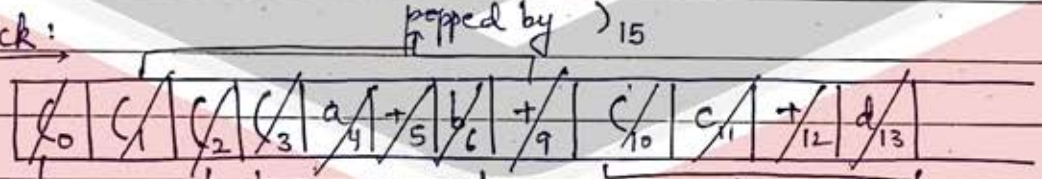
Return the count of duplicate brackets, as pairs ('(')', in the given string.

How?

If '(' → push in stack.
If ')' → pop till you get an '(', pop that '('.
If any other char, push.

ex: (((((a+b))) + (c+d)))
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

stack:



popped by)7
popped by)14
popped by)8 → since this pop had no elements in between ( ), count of duplicate brackets c+=1. (1)
popped by)16,
since count of elements in between was 0, c+=1 (2)

∴ c = 2 ⇒ 2 duplicate bracket pairs were found.

1) 0 – 6 were pushed.
2) 7 popped 3 – 6.
3) 8 popped 2 → count = 1.
4) 9 – 13 were pushed.
5) 14 popped 10 – 13.
6) 15 popped 1 and 9.
7) 16 popped 0 → count = 2

Q. Balanced Brackets

What?

You are given a string havid '[', ']', '(', ')', '{', '}' and other characters. Return if the brackets in the string are balanced or not.

How?

Consider 3 cases of mismatch:

1) When opening & closing are not same.
2) When you iterate entire string and still stack is empty.
3) When closing bracket comes but stack is empty.

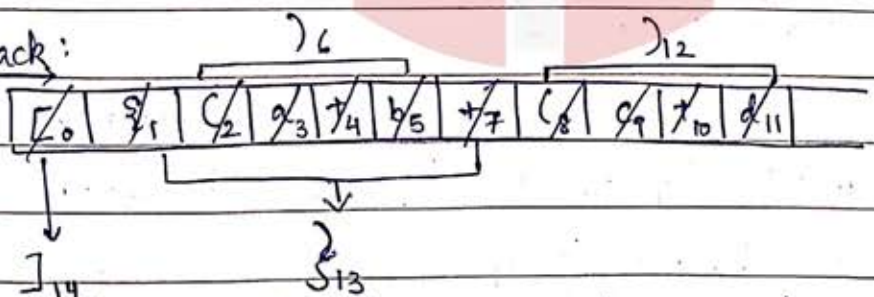
If '}' → pop till '{'
if ']' → pop till '['
if ')' → pop till '('

→ if any of these corresponding closing don't occur, but any other occurs, return false

If in the end stack is empty, return true, else return false.

ex: $[_0\ \{_1\ (_{2}a_{3}+_{4}b_{5})_{6}\ +_{7}\ (_{8}c_{9}+_{10}d_{11})_{12}\ \}_{13}\ ]_{14}$

stack:



In end, stack is empty, return true.

Q. Decoding a string

What?

You are given a string that is encoded, like $k_1[(str1)]k_2[(str2)](str3)\ldots\ldots$ where k_1, k_2 are integers and represent frequency of string in brackets following them. Integers given in string have to be treated as frequencies only. Return decoded string.

How?

ex: 3[a 2[b]] c

= abb abb abb c

→ Use reverse loop (⟵)

If ']' → push,

if character → push

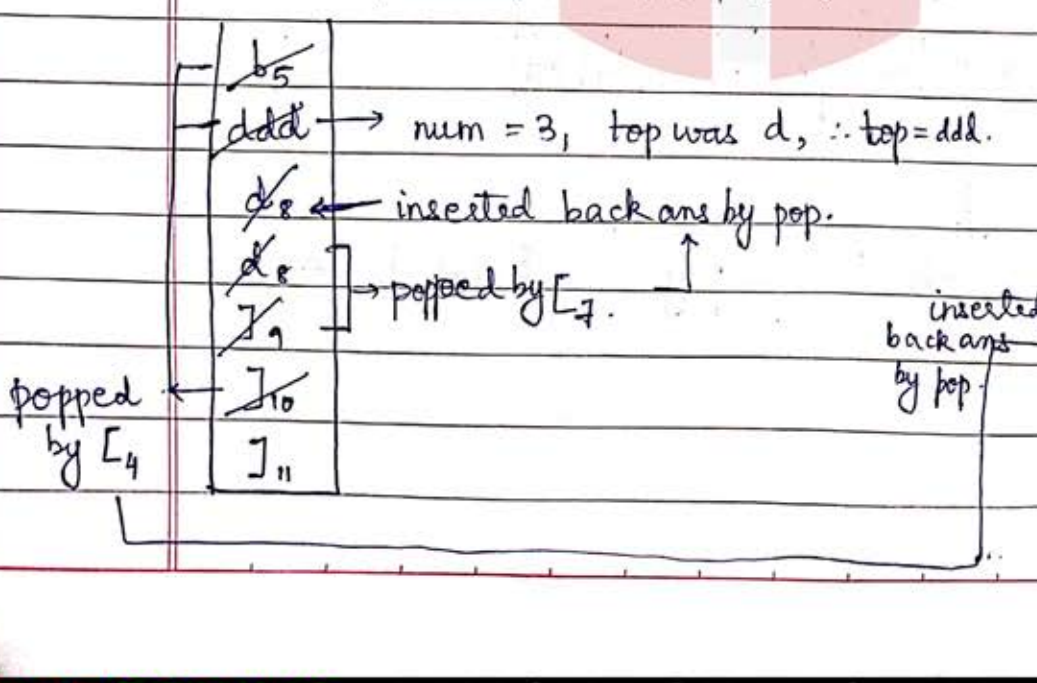
if '[' → settle (pop till ']'), and then push the ans as characters collected while popping.

if number/integer → make top as top * num which means if top is 'd', and num is 3, top becomes 3 times d ⇒ "ddd".

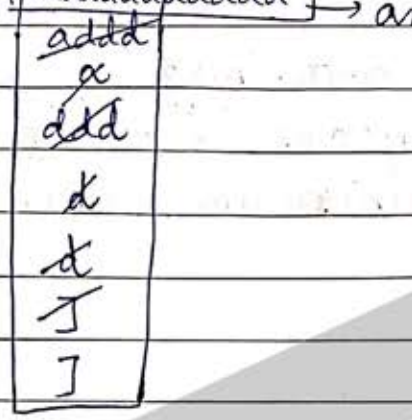
* Stack made is of strings.

ex1: 2 [a 2 [b 3 [d]]]

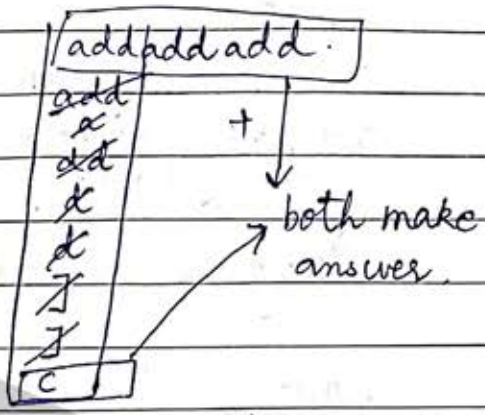
0 1 2 3 4 5 6 7 8 9 10 11



ex2: 3[a 3[d]]



ex3: 3[a2[d]]c.



ans = adddaddaddc

* Make sure that you print answer as the entire stack left after the iteration, because string can be formed by concatenation of multiple string answers.

Q. Bracket Positions

What?

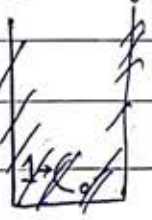
Given an expression of length n consisting of some brackets, print the bracket numbers when the expression is being parsed. Bracket number is position of bracket pair to which current bracket belongs.

How? Keep a counter = 0 variable.

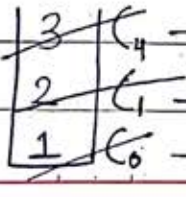
If '(' → counter++, push counter in stack, print.

& if ')' → 1 pop from stack and print it.

ex: $(_0 a + (_1 b + c)_2)_3 + (_4 d + e)_5$



* Stack is of count.



$(_4$ → popped by $)_5$

$(_1$ → popped by $)_2$

$(_0$ → popped by $)_3$

⇒ ans: 1 2 2 1 3 3

↓ ↓ ↓ ↓ ↓ ↓

$(_0$ $(_1$ $)_2$ $)_3$ $(_4$ $)_5$

Q. Number of Reversals

What?

You are given a string containing '{' and '}' only. You can perform reversals means '{' → '}' or '}' → '{'. Perform minimum reversals to make string balanced.

How?

If '{' → push in stack
else if top == '{', pop '{'.
else push '}'.

ex: } } { { } } { { { { { { { {
0 1 2 3 4 5 6 7 8 9 10 11 12 13

stack →

$}_0$	$}_1$	~~$\{_2$~~	~~$\{_3$~~	~~$\{_6$~~	$\{_8$	~~$\{_9$~~	$\{_{11}$	$\{_{12}$	$\{_{13}$.
		$}_5$	$}_4$	$}_7$		$}_{10}$			

Now count no. of opening brackets oc & no. of closing brackets cc.

Here oc = 4, cc = 2.

If oc is odd and cc is also odd or if oc and cc both are even, only then a solution is possible.

$$ans = ceil\left(\frac{oc+1}{2}\right) + ceil\left(\frac{cc+1}{2}\right)$$

$$= 3 + 2 = 5 \text{ reversals.}$$

clearly these are $}_0 \rightarrow \{$ ①, $}_1 \rightarrow \{$ ②
$\{_{11} \rightarrow \}$ ③, $\{_{12} \rightarrow \}$ ④, $\{_{13} \rightarrow \}$ ⑤.

Date
DELTA Pg No.

Q. Making parenthesis valid

What?

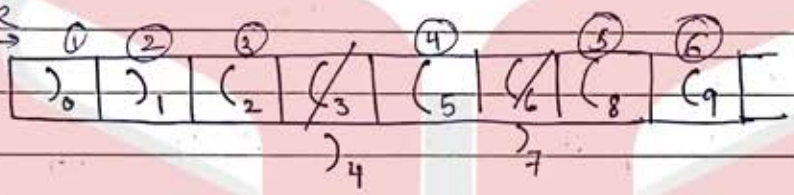
Given a string of '(' and ')', return minimum additions of '(' or ')' to make string valid.
A parenthesis string is valid iff:
a) it is empty ; b) it can be written as AB where A and B are valid ; c) A where A is valid.

How?

If ')' → if top is '(', pop '(', else push ')'.
If '(' → always push.

ex:)) (( ) (( ) ((
0 1 2 3 4 5 6 7 8 9.

stack →



→ ans = 6

After entire iteration, return stack size.

Q. Score of string

What?

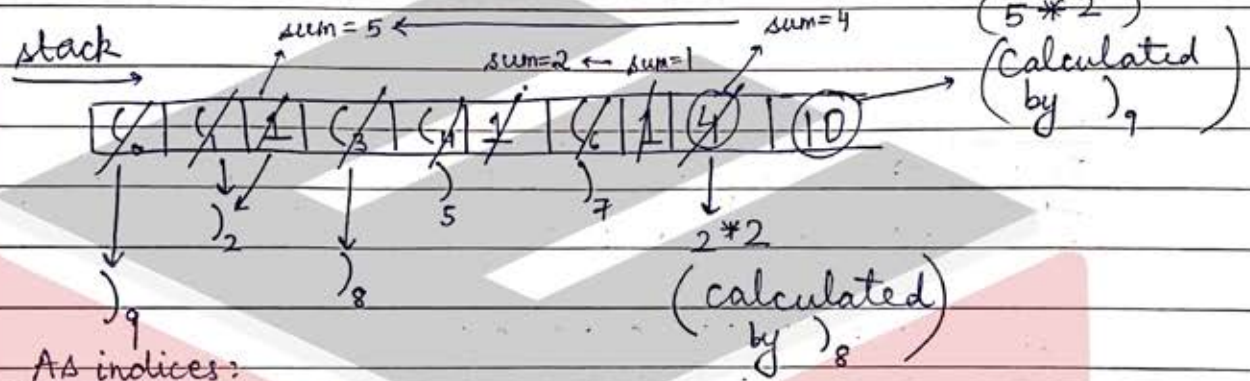
Given a balanced parenthesis string, compute its score as:
() has 1 score, AB has score A+B where A & B are balanced parenthesis strings, (A) has score 2*A where A is balanced parenthesis string.

How?

If '(' → always push.
e if ')' → if top == '(', pop '(' and push 1.

e if ')' → if top != '(',
collect sum as sum += each pop till you get a '(', and then pop it('('), and push 2 * sum.

ex: (( ) (( ) ( )))
0 1 2 3 4 5 6 7 8 9.



As indices:

2) 0, 1 pushed.
2) 2 pops 1, pushes 1 as score of $(_1)_2$
3) 3, 4 pushed.
4) 5 pops 4, pushes 1 as score of $(_4)_5$
6) 6 pushed.
7) 7 pops 6, pushes 1 as score of $(_6)_7$
8) 8 pops result in 7) and 4) to make sum of 2, then pops 3 as $(_3\ 1 + 1\ )_8$ and pushes 4 (2*2).
($(_4)_5$ $(_6)_7$)

9) 9 pops 0 and results of 8) and 2) are added to make sum 5, and then pushes 10 → 5*2.
evaluating: $(_0\ 1 + (_3\ 1+1\ )_8\ )_9$
($(_1)_2$)

Date
DELTA Pg No.

Expressions

Precedence of operators

+ , - < * , / < ^

1) Infix Evaluate.

a) Take 2 stacks, number stack 'ns' and operator stack 'os', for evaluation.

b) If you get a number / integer, push in ns.

c) If '(' or any operator, push in os.

d) If any operator, before pushing in os, first evaluate for expressions where operators have precedence >= current operator.

→ evaluate as : 2 pops from ns, 1 from os, evaluate, result & push result back in ns.

An operator pops till:

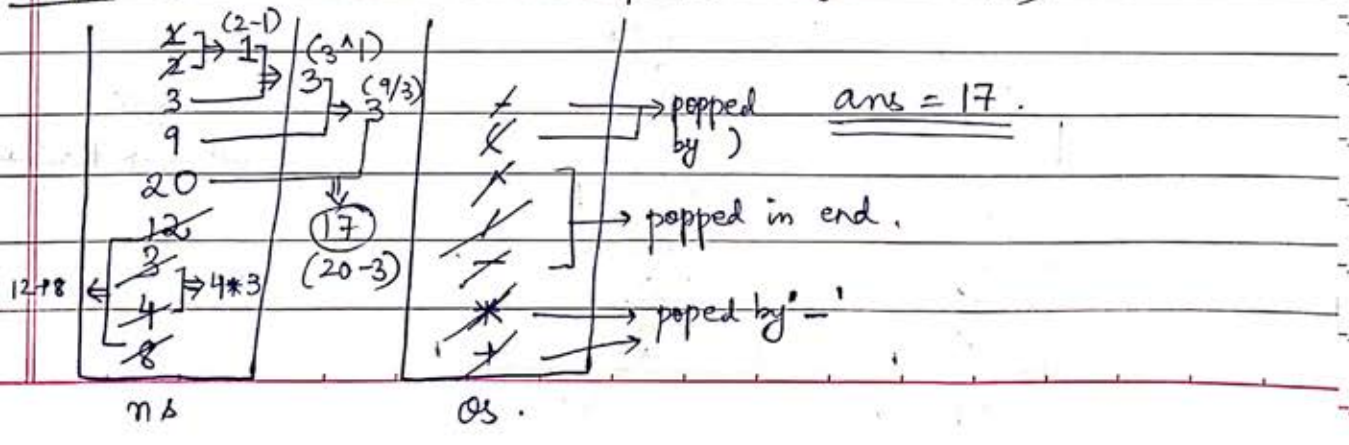
→ stack is empty.

→ top is (.

→ top operator < current operator in precedence.

e) If ')' → pop till '(' and pop '(', for all in-between operator pops, keep evaluating & pushing back result in ns.

ex: 8 + 4 * 3 - 9 / 3 ^ (2 - 1)



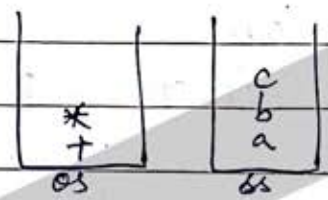
2) Infix → Postfix.

a) Take 2 stacks, string stack and operator stack. ↓ss ↘os

b) If character → push in ss, if operator → push in os.

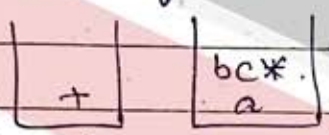
c) If operator → before inserting operator in os, first evaluate results of all operators having precedence >= it.

ex:



Now - comes,

∴ - first makes 2 pops c, b in ss, 1 of * in os, and pushes bc* in ss.



Now + is popped making result abc*-.

The string is calculated as:

pop1 from ~~ss~~ = bc* ①

pop2 from ~~ss~~ = a. ②

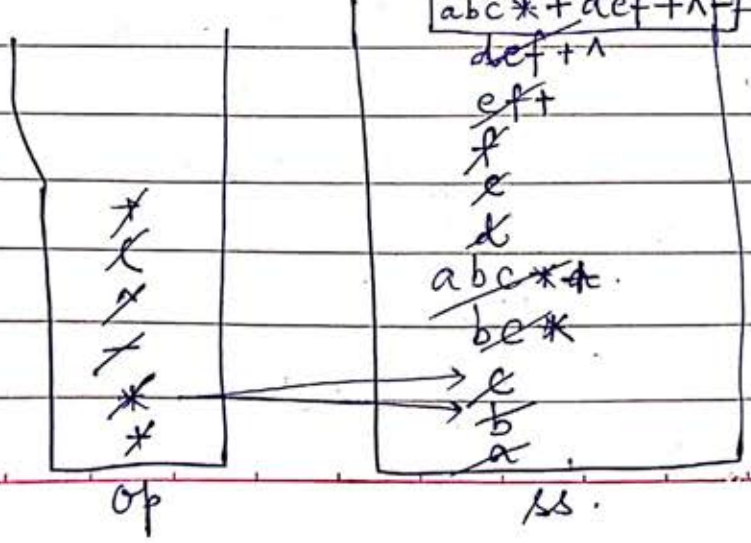
popop from os = +. ③

∴ string res = ② + ① + ③

= a b c * + . (for postfix).

ex: a + b * c - d ^ (e + f).

abc*+def+^- → answer



∴ Postfix:

abc*+def+^-

3. Infix → Prefix

a) Reverse infix expression.

Infix: a+b*c-d^(e+f)

reverse:)f+e(^d-c*b+a

b) Normalise brackets:) → (, (→).

(f+e)^d-c*b+a

c) Now solve postfix for this.

Current ans = fe+d^cb*-a+

d) Now reverse the current ans again.

⇒ Prefix: -+a*bc^d+ef → answer

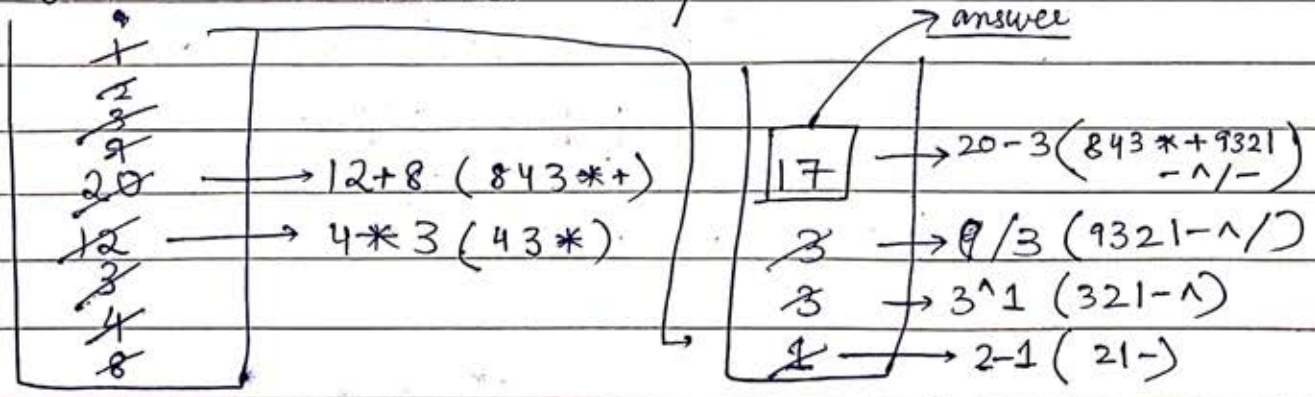
4. Postfix Evaluate

a) If numbers, push.

b) If operators, pop 2 numbers, evaluate the expression, and push back result.

Evaluate as pop2 op pop1.

ex: 8 4 3 * + 9 3 2 1 - ^ / -



5. Postfix → Infix.

a) If numbers, push in stack.

b) If operator, ~~pop all >= precedence operators~~, evaluate result for ~~each~~ this operator, and then push operator.

Evaluate as:

pop1 = first element popped

pop2 = second element popped.

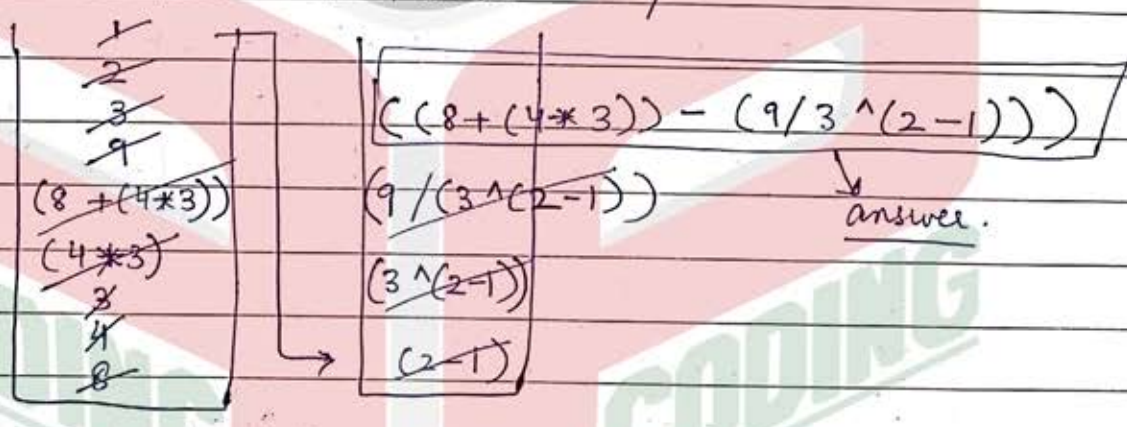
string S = "(" + pop2 + op + pop1 + ")".

↓

where op is current operator

Append S to final result string each time, and add back S in stack.

ex: 8 4 3 * + 9 3 2 1 - ^ / -.

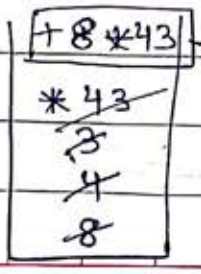


6. Postfix to Prefix.

Just like postfix → infix, only the evaluation for an operator is done as

S = op + pop2 + pop1.

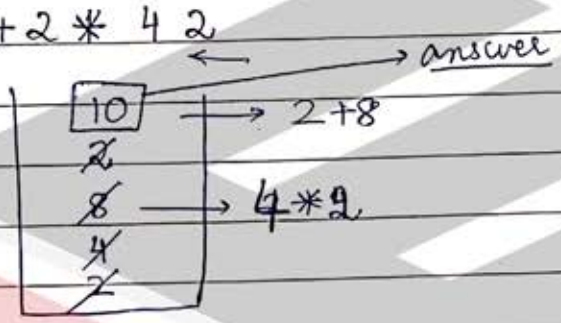
ex: 8 4 3 * +



7. Prefix Evaluate

a) Loop from reverse.
b) If numbers, push.
c) If operators, pop 2 numbers, evaluate the expression, and push back result.
Evaluate as pop1 op pop2.

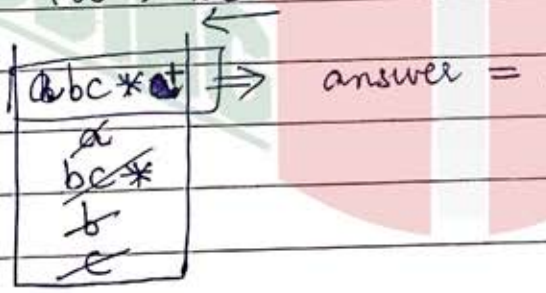
ex: +2 * 4 2



8. Prefix → Postfix.

a) Loop from reverse
b) If characters / numbers, push.
c) If operators, pop 2, evaluate the expression as pop1 + pop2 + op.

ex: +a * bc

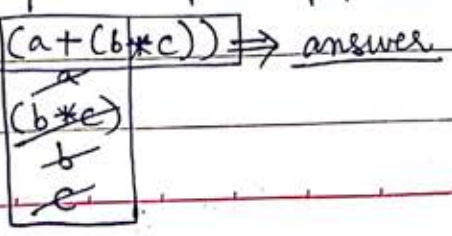


⇒ answer = abc*+

9. Prefix → Infix.

#Same as Prefix → Postfix, just evaluate as
S = "(" + pop1 + op + pop2 + ")".

ex: +a *bc.



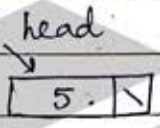
⇒ answer

Stack Adapters

1. Adapt a Linked List to make a stack.

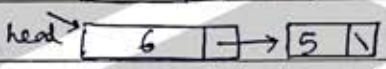
For top : return head.data if head != NULL.
For push : use addFirst in LinkedList.
For pop : use ~~addata~~ removeFirst in LinkedList.

ex: Push 5 :

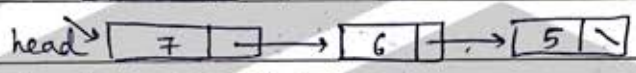


(assign head if empty).

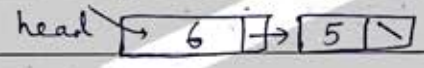
Push 6 :



Push 7 :



Pop :



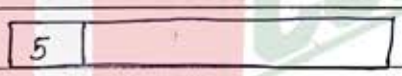
returns 7.

* You can also use addLast and removeLast if you maintain a tail in LinkedList.

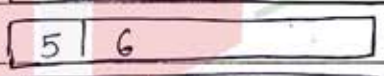
2. Adapt an ArrayList to make a stack.

For top : return list.get(0) i.e., first element.
For push : use addLast in arrayList / vector.
For pop : use removeLast in ArrayList / vector.

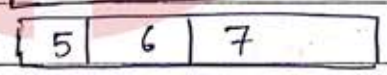
ex: Push 5 :



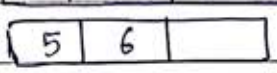
Push 6 :



Push 7 :



Pop :



returns 7.

* Since addFirst and removeFirst are of O(n) in arrayList, we do not use these two.

Date
DELTA Pg No.

3. Adapt 2 queues to make a stack (push efficient)
Take 2 queues, dq is data queue, hq is helper queue.
For push: Simply push data in data queue.

Push 5: dq | 5 |
Push 6: dq | 5 | 6 |
Push 7: dq | 5 | 6 | 7 |

For top: Copy contents of dq in hq as a series of removeFirst from dq and addLast in hq. The last element added in hq is top.

Top: dq | 5 | 6 | 7 |
hq | 5 | 6 | 7 | → return 7

* Now dq = hq & hq = new queue (swap)

For pop: Copy contents just like we do in top, just don't add the last element, return its value.

Pop: dq | 5 | 6 | 7 | → return 7
hq | 5 | 6 |
don't add in hq.

* Now dq = hq & hq = new queue (swap).

Since this adaptation is Push efficient;
Push → O(1)
Pop → O(n)
Top → O(n)

4. Adapt 2 queues to make a stack (Pop efficient)
Take 2 queues, dq is data queue, hq is helper queue.

For push: If dq is empty, push in dq.
Else, add data in hq, now copy all contents from dq to hq as series of removeFirst and addLast. In the end, swap dq=hq & hq=new queue.

Push 5: dq [5]

Push 6: dq [5]
hq [6 | 5]
Swap: dq = [6 | 5], hq = [ ]

Push 7: dq [6 | 5]
hq [7 | 6 | 5]
Swap: dq = [7 | 6 | 5] hq = [ ]

Top: Return first index of queue, that is, return dq.top() / dq.peek().

Top: dq [7 | 6 | 5] → return 7

Pop: dq.removeFirst().

Pop: dq [7 | 6 | 5] → return 7.

dq now is : [6 | 5]

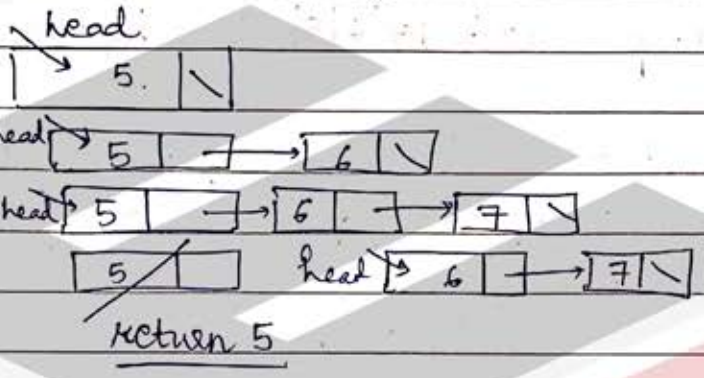
Since this adaptation is Pop efficient,

Push → O(n)
Pop → O(1)
Top → O(1).

Date
DELTA Pg No.

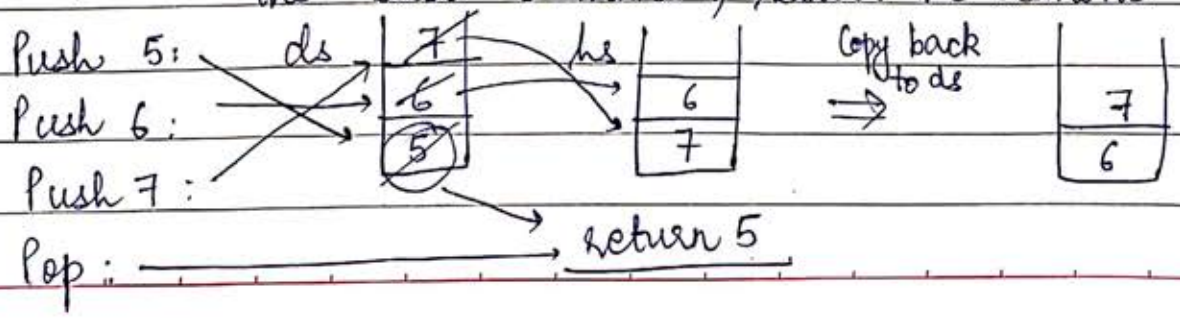
Queue Adapters

1. Adapt a LinkedList to make a queue.
For top: return head.data if head != NULL (peek).
For push: Use addLast to add data. (enqueue)
For pop: Use removeFirst to remove data (dequeue)



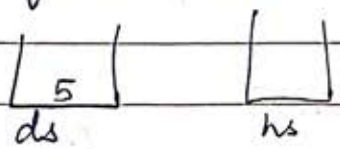
2. Adapt an ArrayList to make a queue.
For top: return list.get(0) or 0th index. (peek).
For push: Using addLast will make pop as addFirst → which is O(n).
and using addFirst for push is O(n).
Hence adapting arrayList to queue formation is not optimal.

3. Adapt 2 Stacks to make a Queue. (Push efficient)
Use 2 stacks, data stack ds and helper stack hs to implement queue.
For push: Simply push data in ds.
For pop: Copy all contents of ds in hs except the last element, return the element.

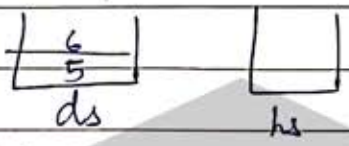


For top: Copy all contents of ds in hs, return top of hs and then copy back in ds.

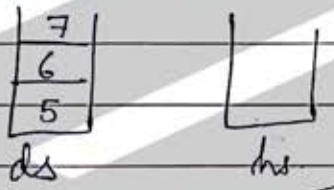
Push 5:



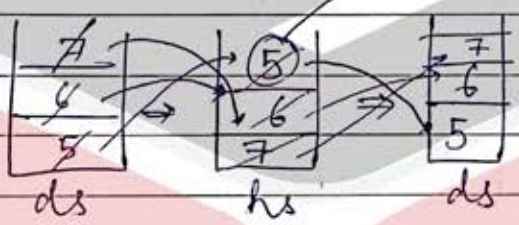
Push 6:



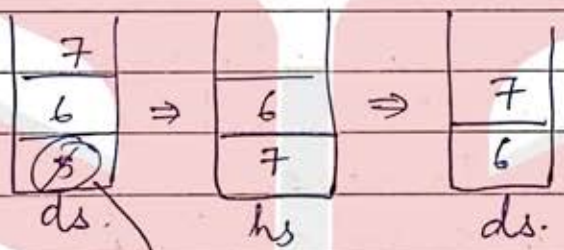
Push 7:



Top:

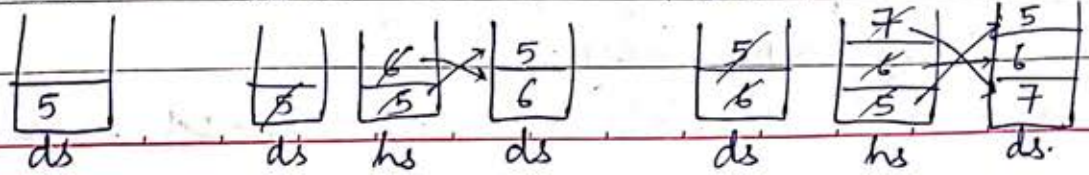


Pop:



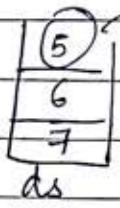
4. Adapt 2 Stacks to make a queue (pop efficient)
Use 2 stacks, data stack and helper stack (ds & hs) to implement queue.
For push: If ds is empty, push in ds,
else, pop all contents of ds in hs,
add data in hs, and then pop all from hs to ds back.

Push 5: Push 6: Push 7:



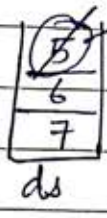
Both pop and top are now the top element of stack. If pop, remove first element; else if top, just return it.

Top:



return 5

Pop:



return 5.

For push efficient:
Push/enqueue → O(1)
Peek & dequeue → O(n)

For pop efficient:
Enqueue → O(n)
Peek & dequeue → O(1)

Q. Design an LRU Cache.

What?

The task is to implement methods of LRU Cache. The class has two methods get() and set().
get(x): Gets the value of key x if the key exists in the cache otherwise returns -1.
set(x, y): Inserts the value if the key x is not already present.
If cache reaches its capacity, it should invalidate the least recently used item before inserting new item.

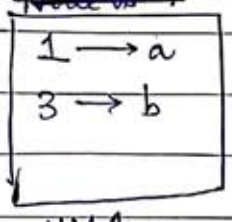
How?

Design implementations.
1) HashMap <Integer, Node> which stores the Node corresponding to an integer value (which is key)
2) HashMap <Integer, Integer> which stores value corresponding key.

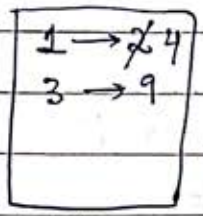
* Node is the Node of double ended linked list.

→ How to perform set (x, y).

~~Node vs Int~~

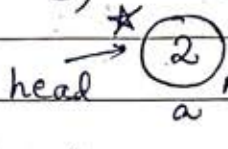


HM1
~~Int vs Node~~



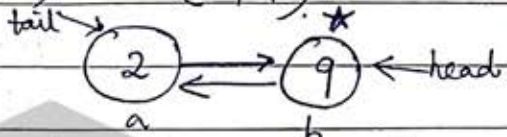
HM2
Int Vs Int

1) Set (1, 2)

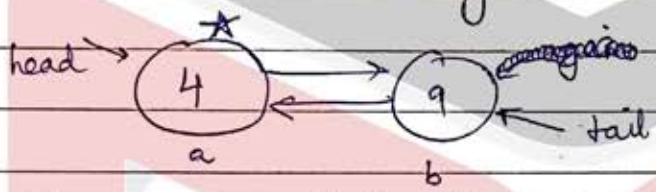


(Let ★ signify Most Recently Used)

2) Set (3, 9)

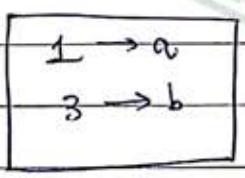


3) Set (1, 4) → since 1 already exists in HM2, update data of 1 to 4 in HM2 and go the node a using HM1 and update data in node.

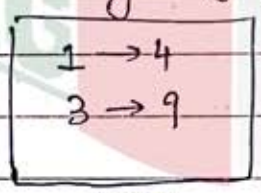


If any set (x, y) is called, the node corresponding to key = x moves towards the front of the dequeue representing that it is MRU.
Tail stores the LRU at any point.

→ How to perform get (x).

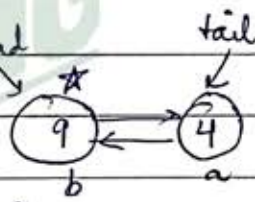


HM1



HM2

1) Get (3)
↳ returns 9.



2) Get (2)
Since 2 does not exist, return -1.

After each get (x), node corresponding to key = x moves at head as now its is MRU.

Algorithm :

Get : get(k)

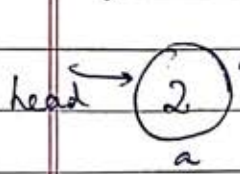
```
{ if (hm1.ck(k))
    { node = hm1.get(k)
      remove node.
      add in front of dequeue
      declare it head.
      return node.data
    }
  else
    { return -1 }
}
```

Set : set(k, v)

```
{ if (hm2.ck(k))
    { if (hm2.size() == cap)
        { discard ~~remove~~ tail
          remove tail entry from both maps.
        }
      Node = new Node()
      add node to front.
      declare head.
      add to both maps.
    }
  else
    { node = hm1.get(k)
      node.data = v;
      remove node, add in front.
      declare head.
    }
}.
```

ex: Cap = 2.

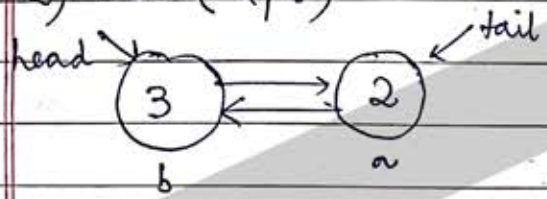
1) Set (1, 2)



HM1
~~1 → a~~
~~2 → b~~
4 → c
6 → d

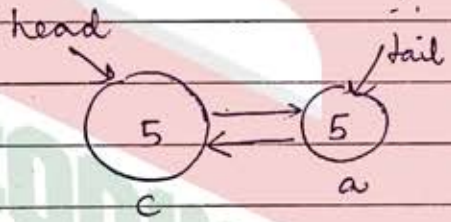
HM2.
~~1 → 2~~ 5
~~2 → 3~~
4 → 5
6 → 7

2) Set (2, 3)

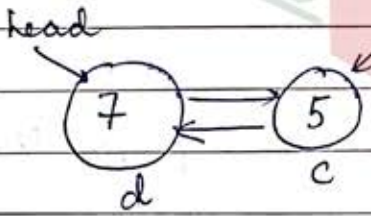


3) Set (1, 5)

4) Set (4, 5) ⇒ here cap = 2 & HM2.size() == 2,
∴ discard 2 → 3 (3)b



5) Set (6, 7) ⇒ here cap = 2 & HM2.size() == 2,
∴ discard 1 → ~~5~~ (5)a.

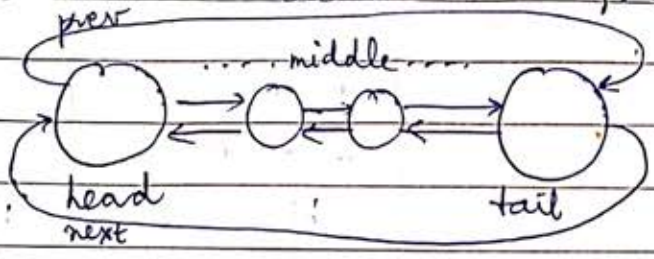


6) Get (4) → returns HM2.get(4) = 5

7) Get (1) → since 1 is neither in HM2, nor HM1, means it was removed.
∴ return −1.

Ans: 5 −1

Q. Circular Double Ended Queue (circular dequeue)



→ Insert a node in ~~between~~ last (addLast)

```
Node n = new Node(data);
n.next = head;  n.prev = tail;  tail = n;
```

→ Insert a node in first (addFirst)

```
Node n = new Node(data);
n.next = head;  n.prev = tail;  head = n;
```

→ Remove a node from last (removeLast)

```
Node n = tail;
tail.prev.next = tail.next;
tail.next.prev = n.prev;
tail = n.prev;
return n.prev = NULL;  n.next == NULL;
return n;
```

→ Remove a node from first (removeFirst)

```
Node n = head;
head.prev.next = head.next;
head.next.prev = tail;
head = head.next;
head.next n.next = NULL;  n.prev = NULL;
return n;
```