11/03/2024

Page No. Date

Day 14 OF DSA.

Talks

Checkbox stack-L

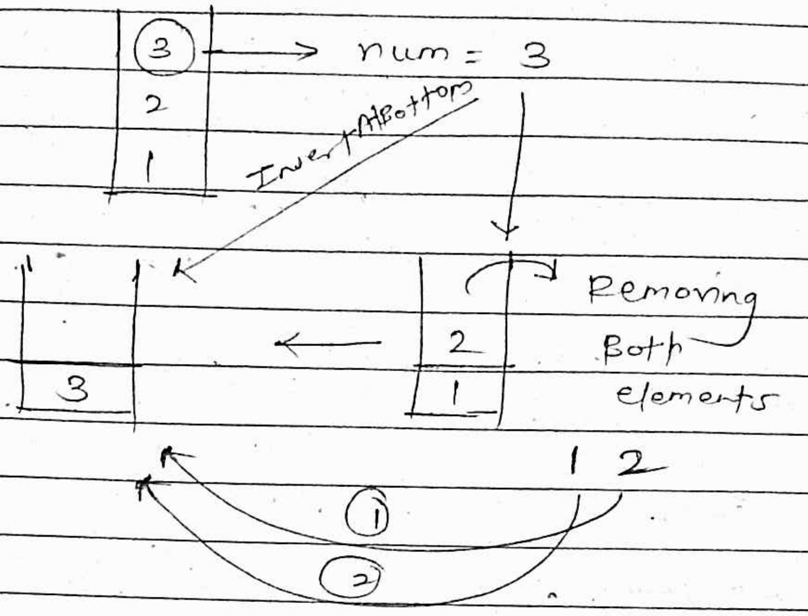
☑ Reverse Stack Using Recursion

☑ Sort stack

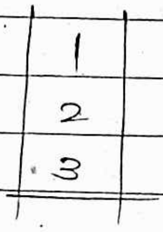
☑ Redundant Brackets

☑ Minimum Cost to Make String Valid

Reverse stack Using Recursion!-



Putting Back into stack



Approach!-

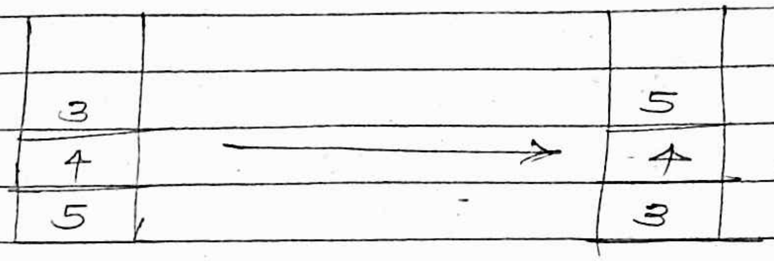
1. Remove 1st element of stack
   Store it in another variable

2. pop all elements &
   the push into stack Using Recursion
   For reversal purpose

3. Then Using InsertAtBottom Function
   push num into that stack

✓

* Sort stack :-



Approach :-

① Remove every element of stack top
& put it in num variable
then pop it.

② Use recursive call to empty() stack

③ Make sortedInsert function.
- In that Insert num in sorted manner with condition
if s.top < num then we push it.

Redundant Brackets

$$\underbrace{(\,(a+b)\,)}_{\text{Redundant Brackets.}}$$

+ Approach :-

1. if we have opening bracket
   or a operator like '+', '-', '*', '/'
   then, we push into stack

2. & in else part
   if we have closing bracket
   then we looking for 'opening bracket in stack
   & pop all elements till "("
   & we False redundant bool variable

3. After loop if we find out
   bool redundant == true
   then we return true
   ~~otherwise~~ False,
   & pop that

Page No.
Date

. 3 types
① {{{{
② }}}} ③ }}}{{{

// Minimum cost to Make string Valid.

{ ( ) }   } {

✓   ✗

* Approach:-

```
(1.) IF string is odd then return -1;

(2) Traverse string & ch = s[i] & stack<char> s;
        check if s.top() == '{'
            then push it

(3) else close braces. '~~{~~' '}'
        & check another if condition if s.top() == '{'
            then pop it,
        otherwise we push into stack

(4) After For loop.
        take two variable a & b
        while string is not empty
            we calculate '{' = b++.
            or —||— '}' = a++;
        & pop every element.

(5) At the end calculate using formular
        ans = $\frac{(a+~~b~~1)}{2} + \frac{(b+1)}{2}$;
```

✓