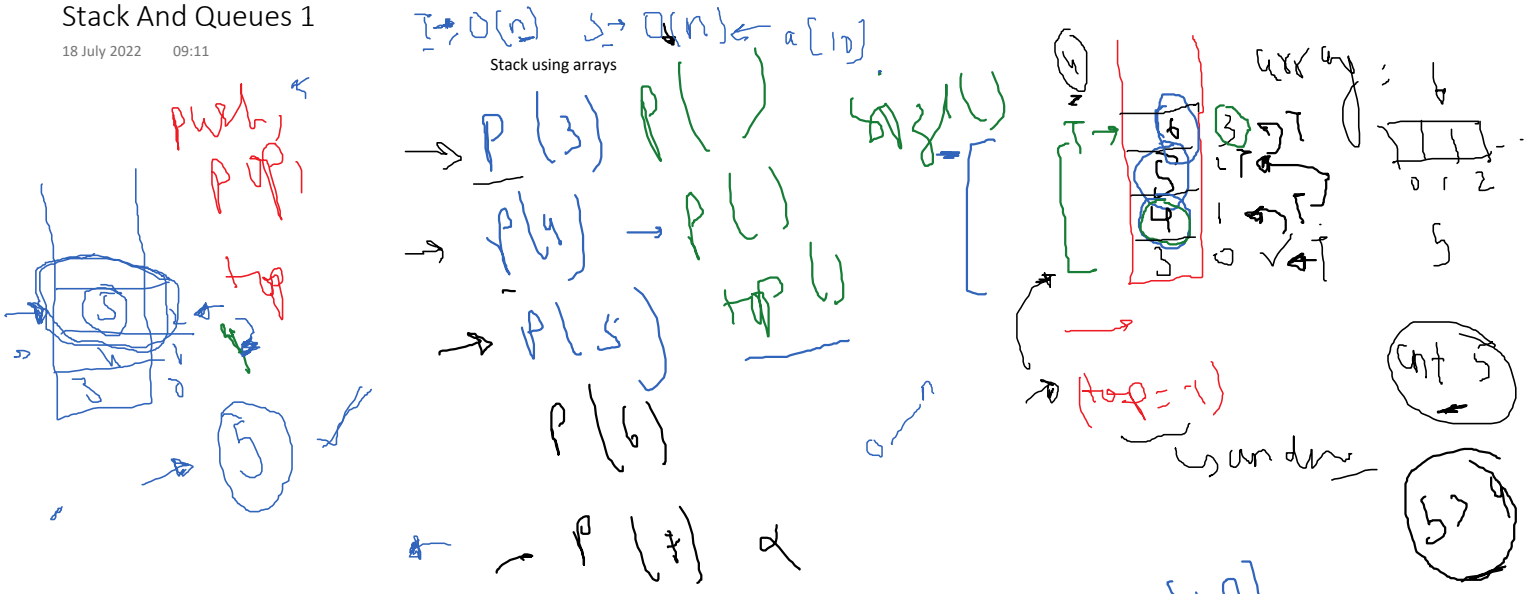
Stack And Queues 1

18 July 2022 09:11

Stack using arrays

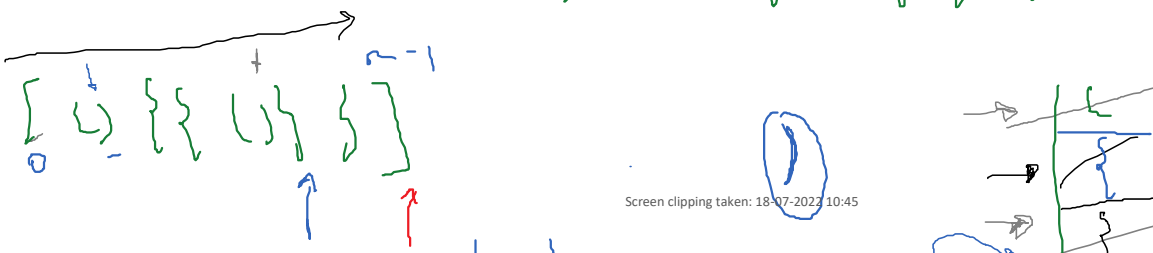


Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

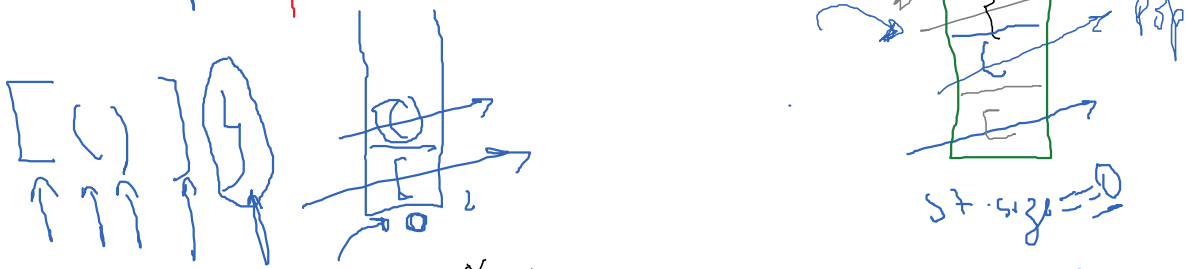
An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

Screen clipping taken: 18-07-2022 10:18

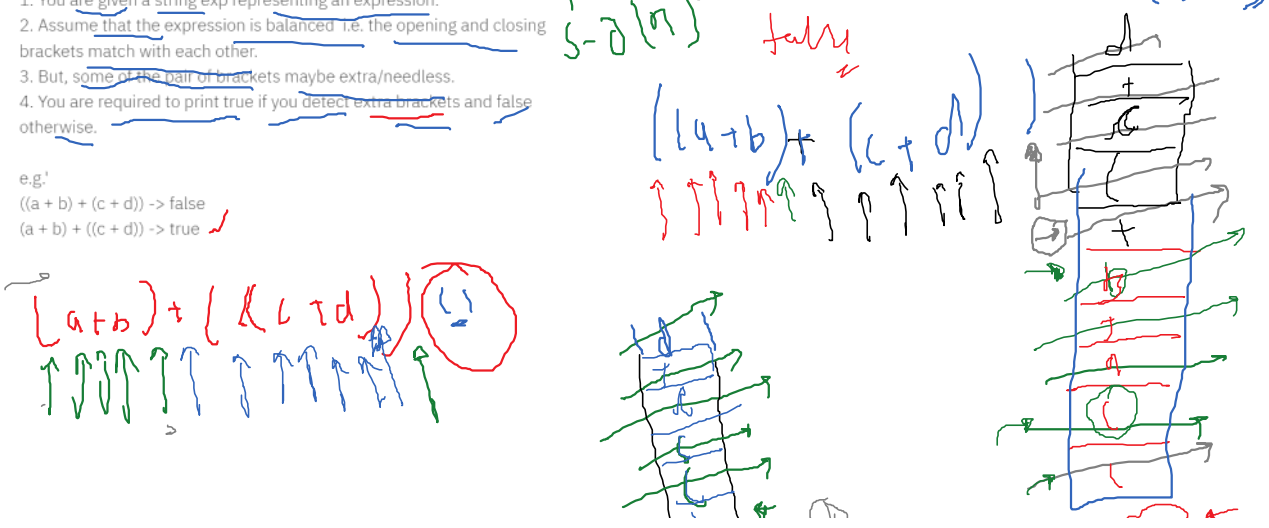


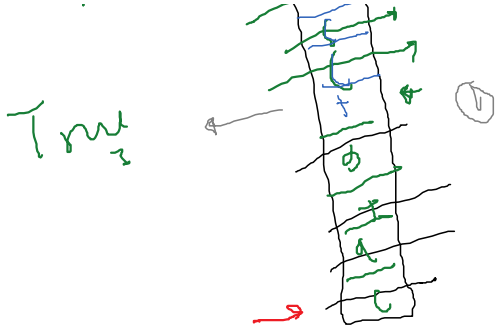
Screen clipping taken: 18-07-2022 10:45

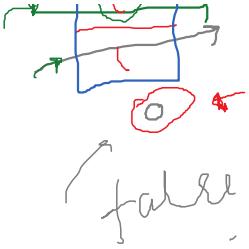


1. You are given a string exp representing an expression.
2. Assume that the expression is balanced i.e. the opening and closing brackets match with each other.
3. But, some of the pair of brackets maybe extra/needless.
4. You are required to print true if you detect extra brackets and false otherwise.

e.g.'
((a + b) + (c + d)) -> false
(a + b) + ((c + d)) -> true



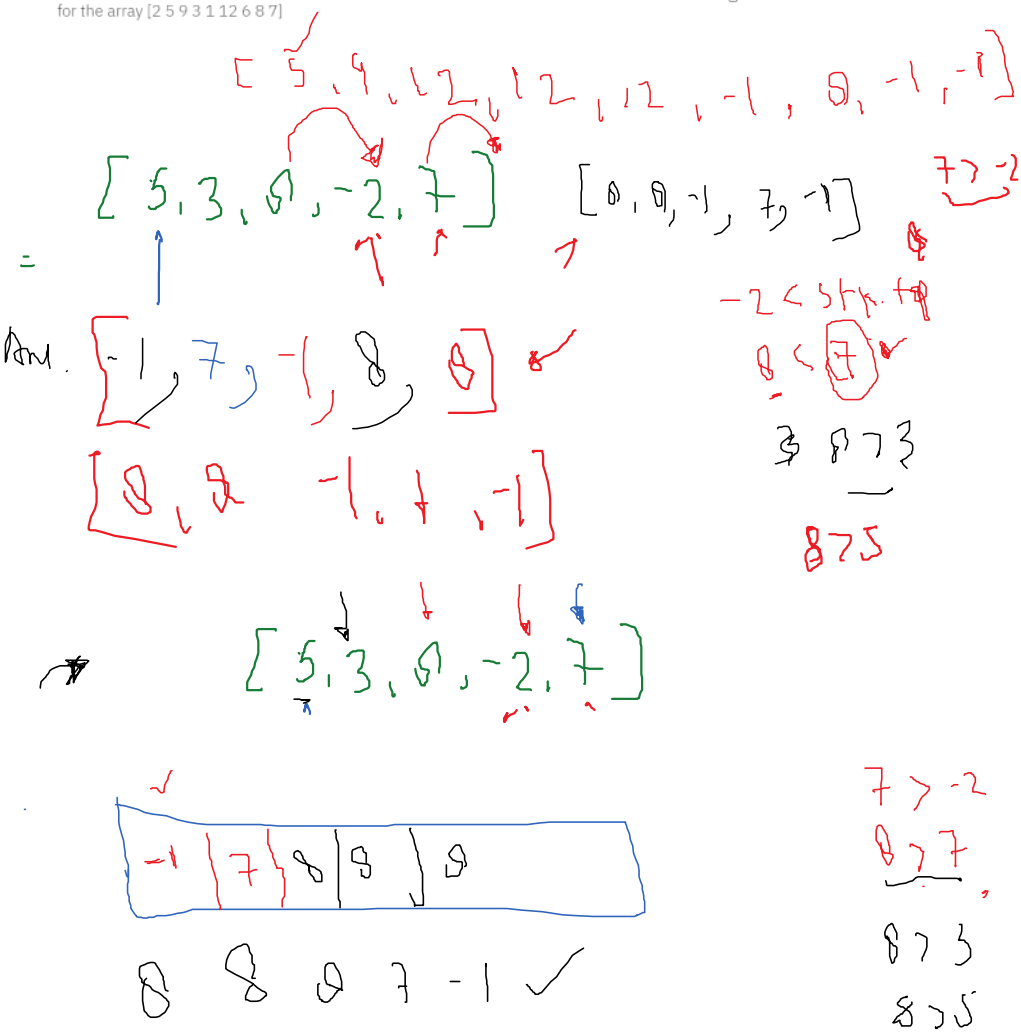


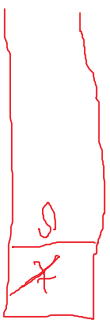


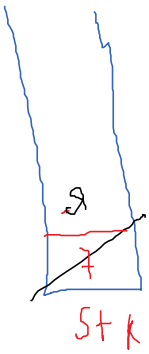
1. You are given a number n, representing the size of array a.
2. You are given n numbers, representing elements of array a.
3. You are required to "next greater element on the right" for all elements of array
4. Input and output is handled for you.

"Next greater element on the right" of an element x is defined as the first element to right of x having value greater than x.
Note -> If an element does not have any element on it's right side greater than it, consider -1 as it's "next greater element on right"
e.g.
for the array [2 5 9 3 1 12 6 8 7]

for the array [2 5 9 3 1 12 6 8 7]
Next greater for 2 is 5
Next greater for 5 is 9
Next greater for 9 is 12
Next greater for 3 is 12
Next greater for 1 is 12
Next greater for 12 is -1
Next greater for 6 is 8
Next greater for 8 is -1
Next greater for 7 is -1

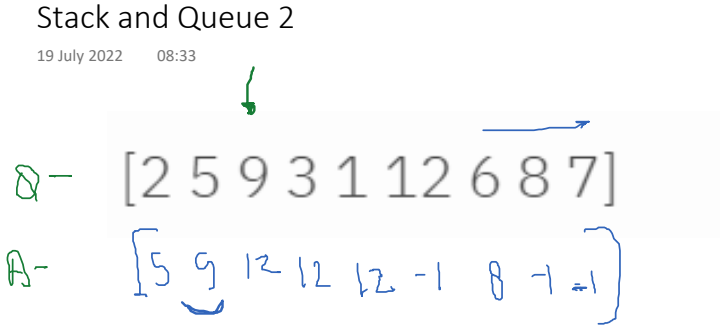


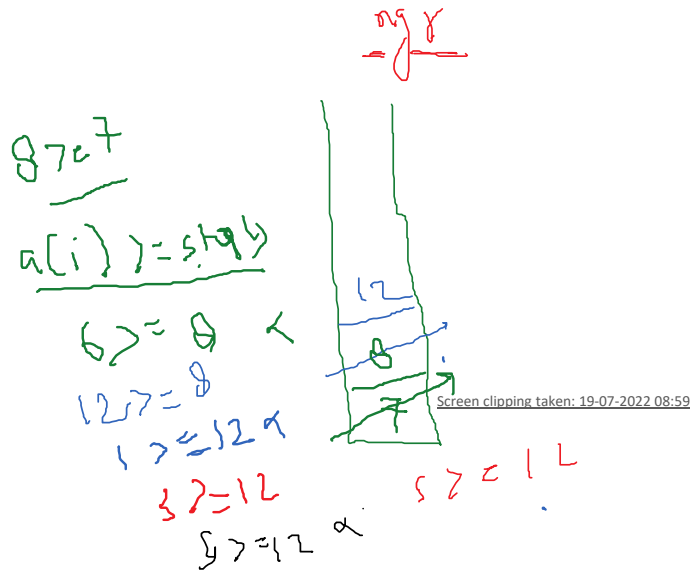


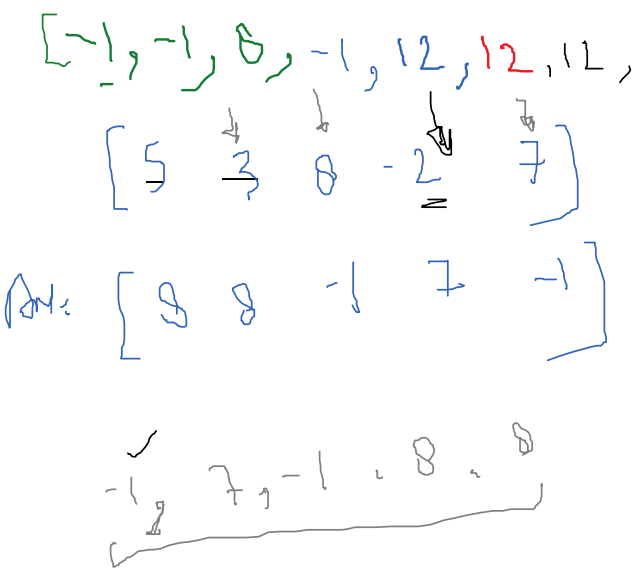


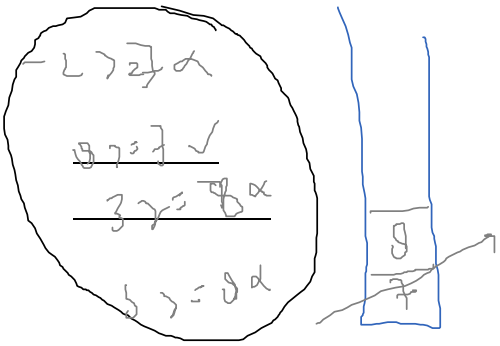
Stack and Queue 2

19 July 2022 08:33









```
public:
//Function to find the next greater element for each element of the array.
#define ll long long
vector<long long> nextLargerElement(vector<long long> arr, int n){

    vector<ll> nge; // stores next greator element
    stack<ll> stk;
    for(ll i =  n - 1 ; i>=0 ; i--)
    {
        while(stk.size() > 0 &&  arr[i] >= stk.top())
        {
            stk.pop();
        }
        if(stk.size() == 0){
            nge.push_back(-1);
        }
        else{
            nge.push_back(stk.top());
        }
        stk.push(arr[i]);
    }
    reverse(nge.begin(), nge.end());
    return nge;
}
```

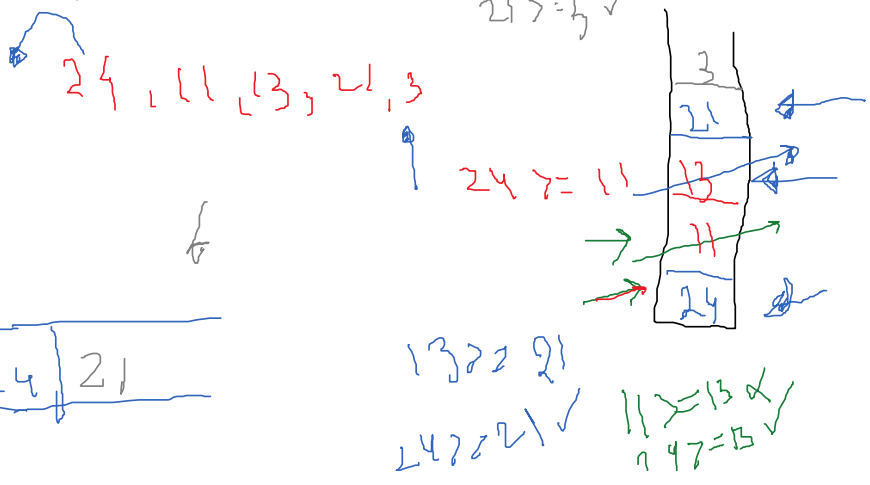
2 5 9 3 1 12 6 8 7

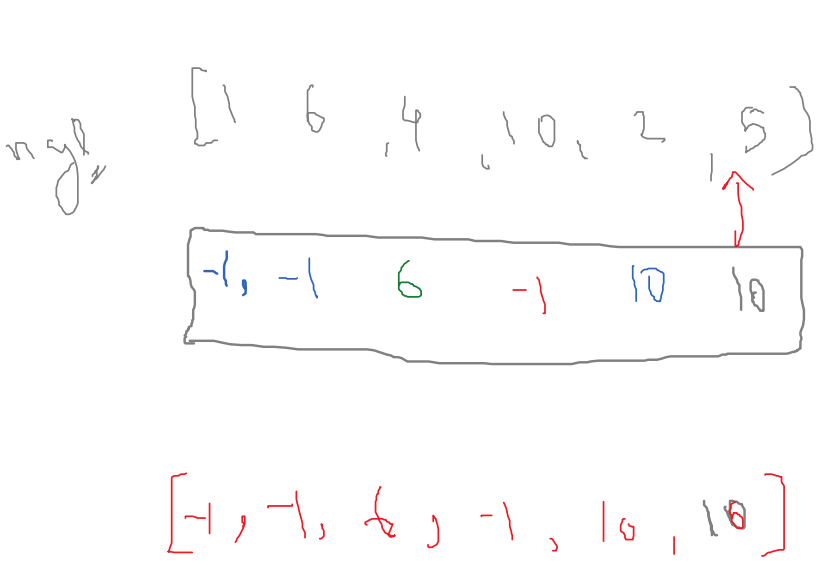
Next Greator element to the Left

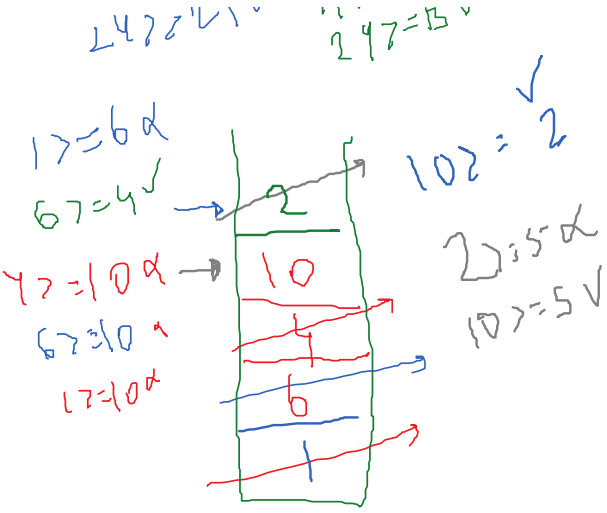
Example -

```
Input: [4, 5, 2, 0]
Output: [-1, -1, 5, 2]

Input: [1, 6, 4, 10, 2, 5]
Output: [-1, -1, 6, -1, 10, 10]
```

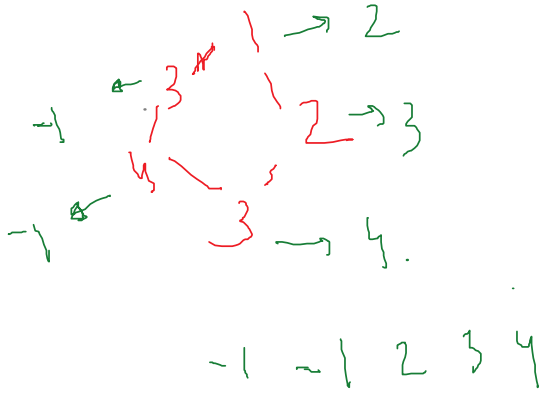






Given a circular integer array `nums` (i.e., the next element of `nums[nums.length - 1]` is `nums[0]`), return *the* ***next greater number*** *for every element in* `nums`.

The **next greater number** of a number `x` is the first greater number to its traversing-order next in the array, which means you could search circularly to find its next greater number. If it doesn't exist, return `-1` for this number.



Example 1:

```
Input: nums = [1,2,1]
Output: [2,-1,2]
Explanation: The first 1's next greater number is 2;
The number 2 can't find next greater number.
The second 1's next greater number needs to search circularly,
which is also 2.
```

Example 2:

```
Input: nums = [1,2,3,4,3]
Output: [2,3,4,-1,4]
```