String Decoding (M to H)

Problem Description

You are given an encoded string **A** of length **N** consisting of digits and lowercase English letters. Your task is to decode the string and return the decoded version.
The encoding rule is as follows:
for every substring in the form of k[encoded_string],
where k is a positive integer and encoded_string is any valid encoded string (it can also include other encoded substrings), you need to repeat the encoded_string exactly k times.

A = 3 [abc] 2 [dj]   ans = abc abc abcdjdj

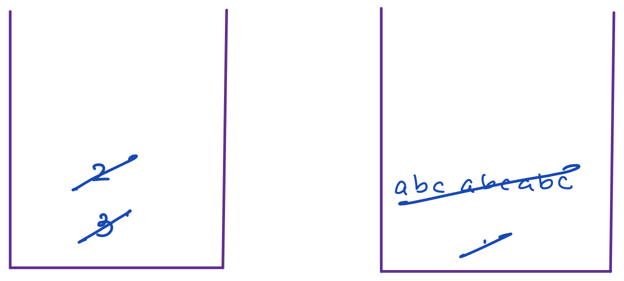
A = 3 [2 [a] bc]   ans = aabc aabc aabc

A = 2 [a] 3 [b]   ans = aabbb

A = 2 [3 [a] c]   ans = aaac aaac

↓

A = 3 [abc] 2 [dj]



nst ↓ num stack

sst ↓ string stack

number → add to nst

[→ add res to sst, res=""

] → calculation

a to z → add ch to res string

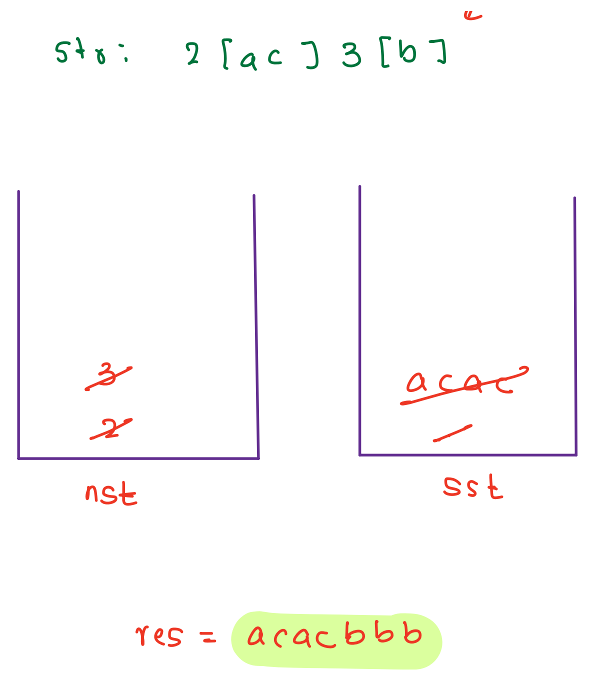
res= abcabcabcdjdj

ch is digit
create no. then add to num stack

ch is [
sst . push (res)
res = " "

ch is]
int count = nst . pop()
temp = old ans + count copies of res
res = temp

ch is a-z
add ch to res string



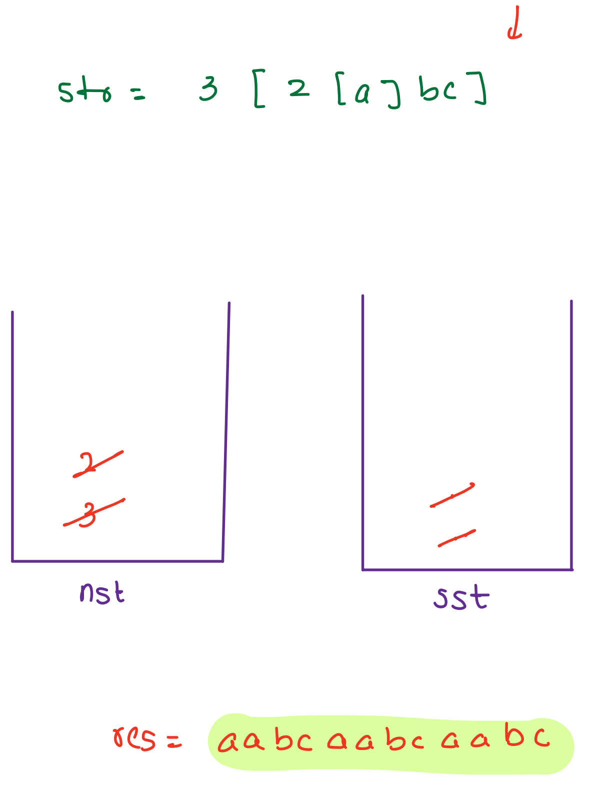
---

ch is digit
create no. then add to num stack

ch is [
sst . push (res)
res = " "

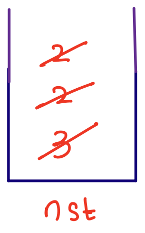
ch is]
int count = nst . pop()
temp = old ans + count copies of res
res = temp

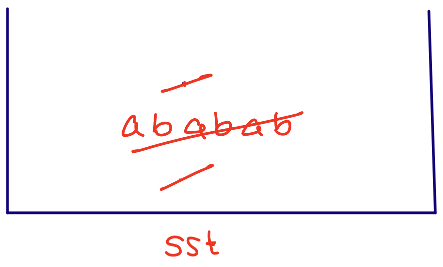
ch is a-z
add ch to res string



```
int i=0;
while(i < A.length()) {
    char ch = A.charAt(i);

    if(isDigit(ch)) {
        int num = 0;
        while(isDigit(A.charAt(i))) {
            num = num * 10 + (A.charAt(i) - '0');
            i++;
        }
        nst.push(num);
    }
    else if(ch == '[') {
        //settling past answer
        sst.push(res);

        //set res = ""
        res = "";

        i++;
    }
    else if(ch == ']') {
        StringBuilder temp = new StringBuilder(sst.pop());
        int count = nst.pop();
        //to temp append count copies of res
        for(int k=1; k <= count;k++) {
            temp.append(res);
        }

        res = temp.toString();
        i++;
    }
    else if(ch >= 'a' && ch <= 'z') {
        res += ch;
        i++;
    }
}
return res;
```

3 [ab] 2 [2[c]mb]





res = abababccmbccmb

```java
public class Solution {
    boolean isDigit(char ch) {
        if(ch >= '0' && ch <= '9') {
            return true;
        }
        else {
            return false;
        }
    }

    public String solve(String A) {
        Stack<Integer>nst = new Stack<>();
        Stack<String>sst = new Stack<>();
        String res = "";


        int i=0;
        while(i < A.length()) {
            char ch = A.charAt(i);

            if(isDigit(ch)) {
                int num = 0;
                while(isDigit(A.charAt(i))) {
                    num = num * 10 + (A.charAt(i) - '0');
                    i++;
                }
                nst.push(num);
            }
            else if(ch == '[') {
                //settling past answer
                sst.push(res);

                //set res = ""
                res = "";

                i++;
            }
            else if(ch == ']') {
                StringBuilder temp = new StringBuilder(sst.pop());
                int count = nst.pop();
                //to temp append count copies of res
                for(int k=1; k <= count;k++) {
                    temp.append(res);
                }

                res = temp.toString();
                i++;
            }
            else if(ch >= 'a' && ch <= 'z') {
                res += ch;
                i++;
            }
        }
        return res;
    }
}
```

Warmer Temperature (M) { next greater on right (index based)}

Problem Description

You are given an array **A** of daily temperatures of **N** days, where **A[i]** represents the temperature on the **i-th** day.
Your task is to find the minimum number of days you have to wait after each day until you can see a warmer day.
If there is no future day for which this is possible, put 0 instead.

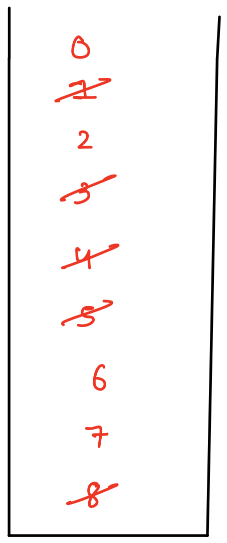
	0	1	2	3	4	5	6	7	8
n =	[24,	21,	28,	20,	19,	25,	29,	32,	30]
nge	2	2	6	5	5	6	7	-1	-1
ans	2	1	4	2	1	1	1	0	0

ans[i] = nge[i] - i

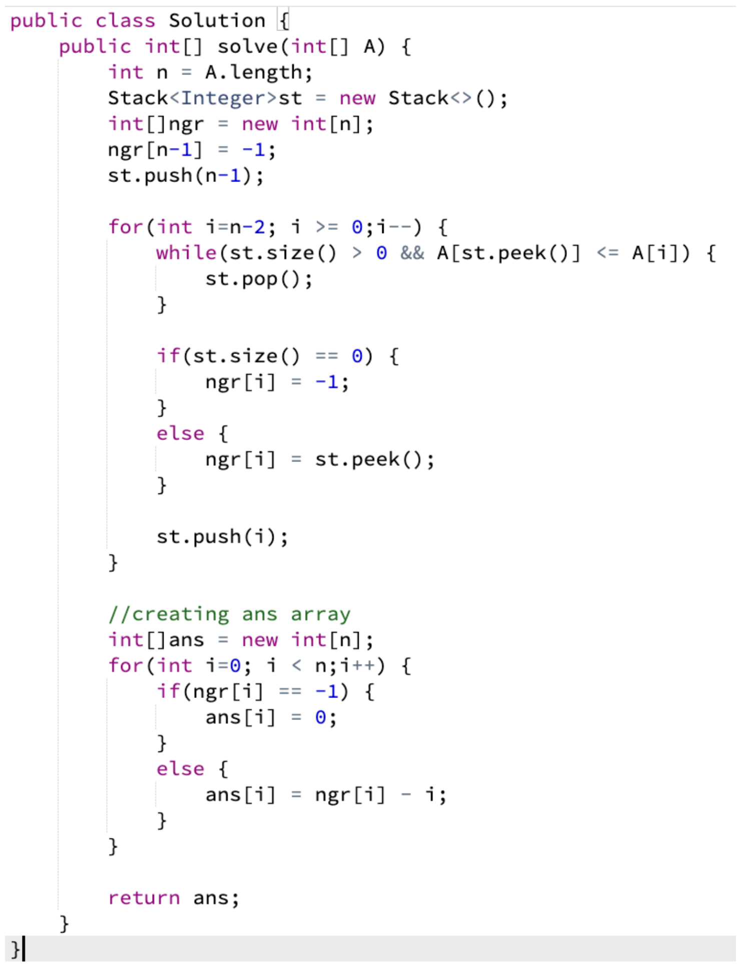
$\swarrow$

	0	1	2	3	4	5	6	7	8
A =	[24,	21,	28,	20,	19,	25,	29,	32,	30]
nge :	2	2	6	5	5	6	7	-1	-1

| A[st.peek()] <= A[i]
  → pop



Index
=



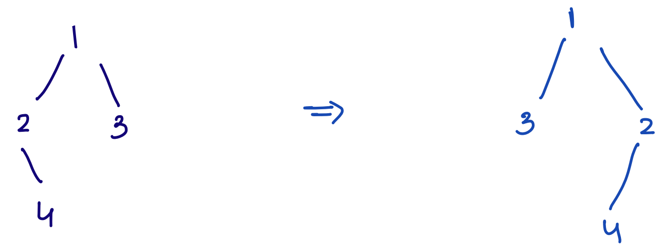
```java
public class Solution {
    public int[] solve(int[] A) {
        int n = A.length;
        Stack<Integer>st = new Stack<>();
        int[]ngr = new int[n];
        ngr[n-1] = -1;
        st.push(n-1);

        for(int i=n-2; i >= 0;i--) {
            while(st.size() > 0 && A[st.peek()] <= A[i]) {
                st.pop();
            }

            if(st.size() == 0) {
                ngr[i] = -1;
            }
            else {
                ngr[i] = st.peek();
            }

            st.push(i);
        }

        //creating ans array
        int[]ans = new int[n];
        for(int i=0; i < n;i++) {
            if(ngr[i] == -1) {
                ans[i] = 0;
            }
            else {
                ans[i] = ngr[i] - i;
            }
        }

        return ans;
    }
}
```

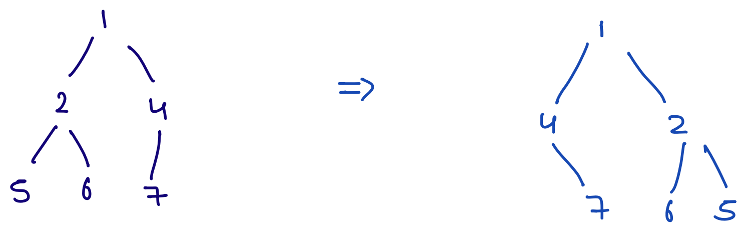
Tree Inversion (M) { travelling the tree }

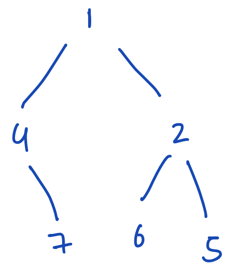
Problem Description

Given the root of a binary tree containing **N** nodes, invert the tree i.e. swap the left and right subtrees and return its root.

↳ mirror image

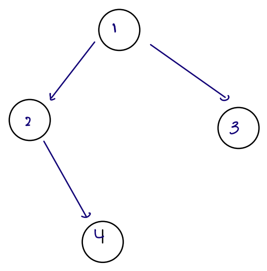


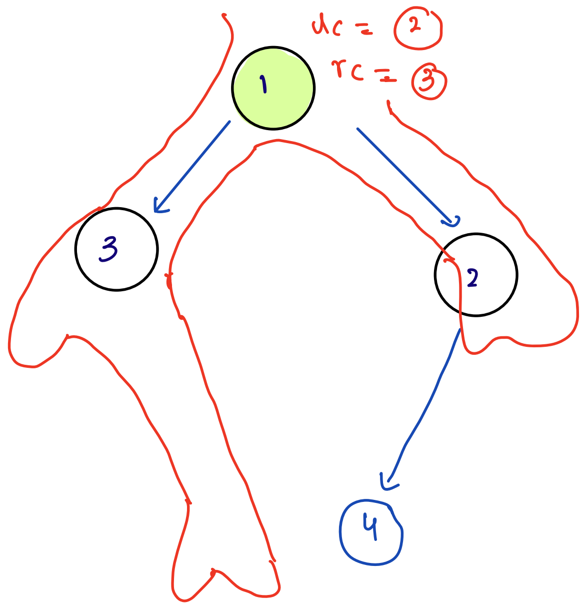




```
TreeNode lc = solve (node.left);

TreeNode rc = solve (node.right);
// swap node's left & right child
node.left = rc ; node.right = lc;
return node;
```

```java
public class Solution {
    public TreeNode solve(TreeNode node) {
        if(node == null) {
            return null;
        }

        TreeNode lc = solve(node.left);
        TreeNode rc = solve(node.right);

        //swap node's left & right child
        node.left = rc;
        node.right = lc;

        return node;
    }
}
```

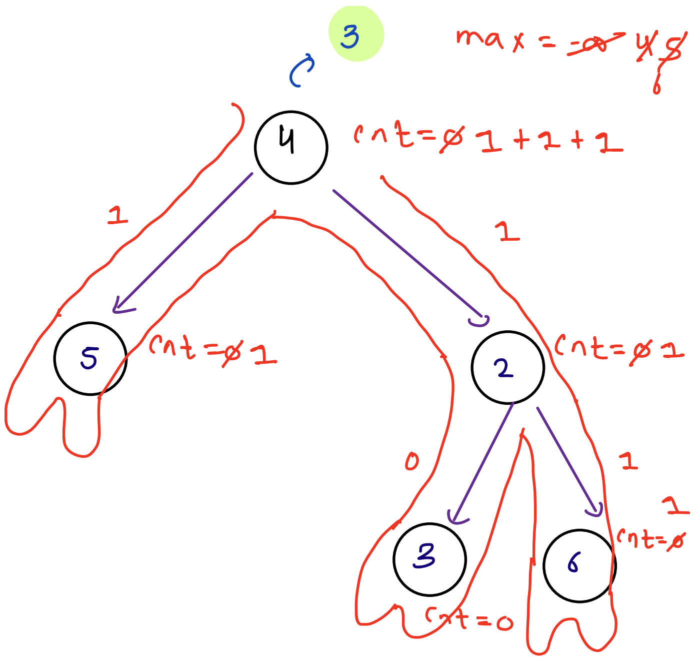




todo: preorder

Doubts

```
int  countNodes (Node  node) {
    int  cnt = 0;
    if  (node == null) {
          return 0;
    }
    if(node.val  >  max)  {
         cnt++;
         max = node.val;
    }

    cnt += countNodes(node.left);

    cnt += countNodes(node.right);

    return  cnt;
}
```



✓ all nodes travelled before node in preorder