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.

Eg: n[String] = [String][String][String]....[String]

Eg = 3[a]2[bc] → aaabcbc

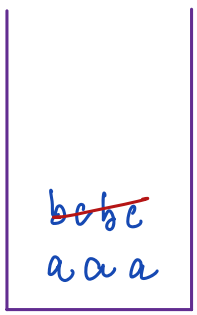
Eg = 2[3[a]b] → 2[aaab] = aaabaaab

Idea: Stack:

Eg = 3[a]2[bc] →

String = a, number = 3 } = aaa

String = bc, number = 2 } = bcbc



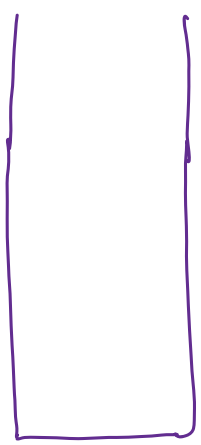
'j': pop till we get open '['
: pop number
: when we insert a string
: if top is already a string
concate string to top string.

Finally: Pop stack till empty & append at start.
= Pop stack ans = aaabcbc

Eg = 2[3[a]b]

String = a, number = 3 } = aaa

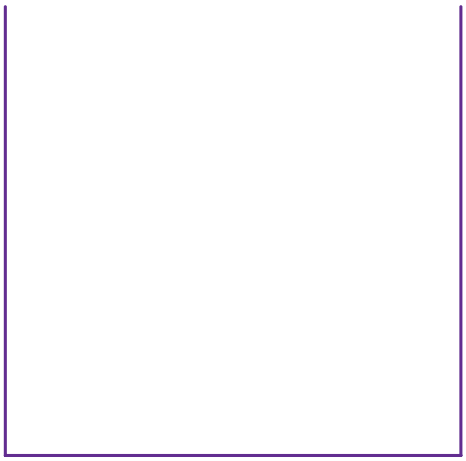
String = aaab, number = 2 } = aaabaaab



[

Ex:

2 [2 [a b] 2 [c a] 3 [d]]

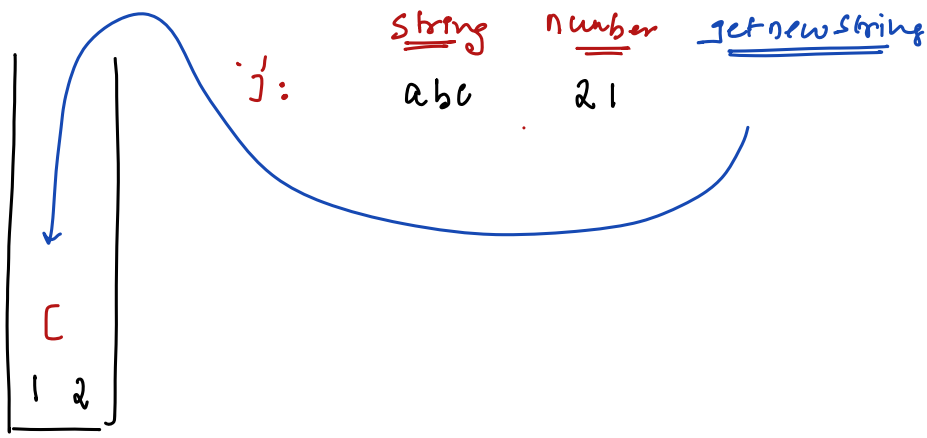


	string	number	
1st:] →	ab	2	→ abab
2nd:] →	ca	2	→ caca
3rd:] →	d	3	→ ddd
4th:] →	abab caca ddd	2	→

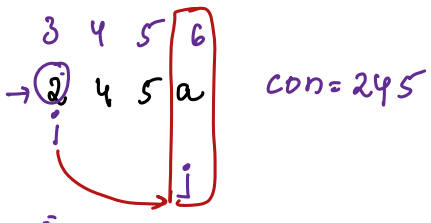
abab caca ddd abab caca ddd

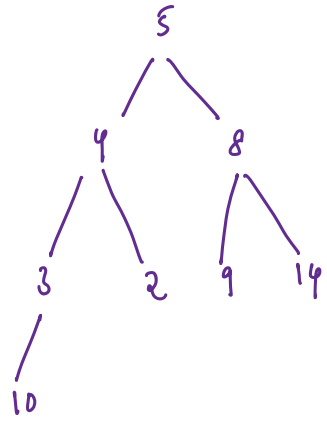
0 1 2 3 4 5 6 7 8 9 10 11 12

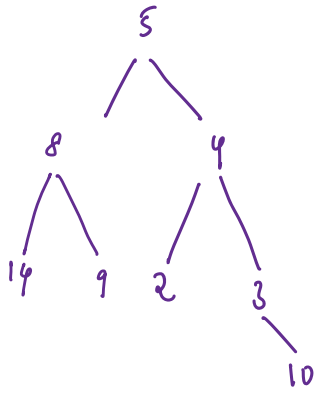
Ex: 1 2 [2 1 [a b c] c d]

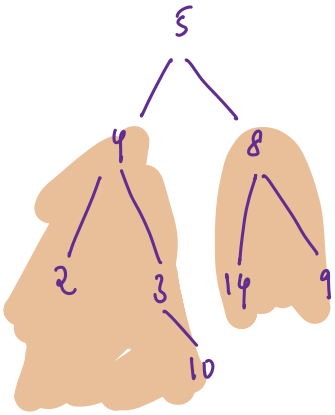


```
string encode(String s){
    Stack<String> st;
    int N = s.length;
    int i=0;
    while(i<N){
        if(s[i]=='['){ st.push(s[i]); i++; } // pushing '['
        else if( s[i]>=48 && s[i]<=57){
            // At ith pos we have number
            String con = "";
            int j=i;
            while(s[j]>=48 && s[j]<=57){
                con = con + s[j]
                j++
            }
            st.push(con);   i=j;
        }
        else if( s[i]>=97 && s[i]<=122){
            String con = "";
            int j=i;
            while(s[j]>=97 && s[j]<=122){
                con = con + s[j]
                j++
            }
            st.push(con); i=j;
        }
        else{ ']'
            1. Keep poping String = & add them to newString = con at start
            2. get number = n
            3. Repeat newString con n times & final string in stack.
        }
    }
}
```

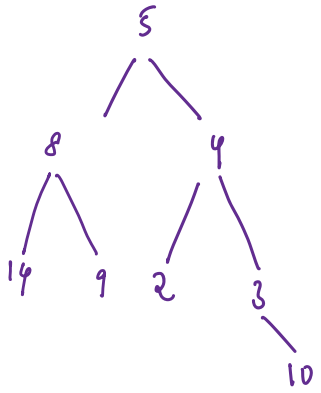








swap left & right child →



	0	1	2	3	4
En: ar[5] =	6	5	8	9	2
nearsgr[ ] =	2	2	3	-1	-1
dist[ ] =	2	1	1	0	0