394. Decode String

Given an encoded string, return its decoded string.

The encoding rule is: k[encoded_string], where the encoded_string inside the square brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer.

You may assume that the input string is always valid; there are no extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there will not be input like 3a or 2[4].

The test cases are generated so that the length of the output will never exceed 10^5.

Example 1:

Input: s = "3[a]2[bc]"

Output: "aaabcbc"

Example 2:

Input: s = "3[a2[c]]"

Output: "accaccacc"

Example 3:

Input: s = "2[abc]3[cd]ef"

Output: "abcabccdcdcdef"

Constraints:

- 1 <= s.length <= 30
- s consists of lowercase English letters, digits, and square brackets '[]'.
- s is guaranteed to be **a valid** input.
- All the integers in s are in the range [1, 300].

```
class Solution {
public:
    string decodeString(string s) {
        const int m = s.size();
        string ans = "",temp = "" ;
        int mul = 0;
        stack<int> num;
        stack<string> st;
        for(int i=0;i<m;i++){
            if(s[i]>='0' && s[i]<='9'){
                mul = mul*10+(s[i]-'0');
            }else if(s[i] == '['){
                st.push(ans);
                num.push(mul);
                mul = 0;
                ans="";
            }else if(s[i] == ']'){
                temp = ans;
                for(int j=1;j<num.top();++j){
                    ans = ans+temp;
                }
                num.pop();
                ans =st.top()+ans;
                st.pop();
            }else{
                ans.push_back(s[i]);
            }
        }
        return ans;
    }
};
```