# 题目

给定一个经过编码的字符串，返回它解码后的字符串。

编码规则为: k[encoded\_string]，表示其中方括号内部的 encoded\_string 正好重复 k 次。注意 k 保证为正整数。

你可以认为输入字符串总是有效的；输入字符串中没有额外的空格，且输入的方括号总是符合格式要求的。

此外，你可以认为原始数据不包含数字，所有的数字只表示重复的次数 k ，例如不会出现像 3a 或 2[4] 的输入。

示例 1：

输入：s = "3[a]2[bc]"

输出："aaabcbc"

示例 2：

输入：s = "3[a2[c]]"

输出："accaccacc"

示例 3：

输入：s = "2[abc]3[cd]ef"

输出："abcabccdcdcdef"

示例 4：

输入：s = "abc3[cd]xyz"

输出："abccdcdcdxyz"

提示：

1 <= s.length <= 30

s 由小写英文字母、数字和方括号 '[]' 组成

s 保证是一个 有效 的输入。

s 中所有整数的取值范围为 [1, 300]

# 分析

思路：

可以使用一个栈来解决这个问题。遍历输入字符串，遇到数字时将数字入栈，遇到字母时将字母入栈，遇到左括号时将左括号入栈，遇到右括号时将栈顶的元素依次出栈直到遇到左括号，并根据出栈的元素构建重复次数和重复字符串，然后将重复后的字符串再次入栈。最后将栈中的元素依次出栈并拼接起来即可得到解码后的字符串。

代码：

class Solution {

public:

string decodeString(string s) {

stack<int> numStack; // 用于存放重复次数的栈

stack<string> strStack; // 用于存放重复字符串的栈

string curStr = ""; // 当前正在构建的字符串

int curNum = 0; // 当前的重复次数

for (char c : s) {

if (isdigit(c)) { // 如果是数字，更新重复次数

curNum = curNum \* 10 + (c - '0'); // 如果是数字25，这里处理这种情况

} else if (isalpha(c)) { // 如果是字母，将其加入当前字符串

curStr += c;

} else if (c == '[') { // 如果是左括号，将当前重复次数和当前字符串入栈，并重置重复次数和当前字符串

numStack.push(curNum);

strStack.push(curStr);

curNum = 0;

curStr = "";

} else if (c == ']') { // 如果是右括号，取出栈顶的重复次数和字符串，并构建重复后的字符串

int repeatTimes = numStack.top();

numStack.pop();

string repeatedStr = "";

for (int i = 0; i < repeatTimes; i++) {

repeatedStr += curStr;

}

curStr = strStack.top() + repeatedStr; // 将重复后的字符串与前缀字符串拼接

strStack.pop();

}

}

return curStr; // 返回最终的解码字符串

}

};