1.面试题 03.04. 化栈为队

题目链接

```
    Stack<Integer> inStack = new Stack();
    Stack<Integer> outStack = new Stack();

    /**
     * Initialize your data structure here.
     */
    public MyQueue() {

    }

    /**
     * Push element x to the back of queue.
     */
    public void push(int x) {
        inStack.push(x);
    }

    public void transfer() {
        if (!outStack.isEmpty()) return;
        while (!inStack.isEmpty()) {
            outStack.push(inStack.pop());
        }
    }

    /**
     * Removes the element from in front of queue and returns that element.
     */
    public int pop() {
        transfer();
        return outStack.pop();
    }

    /**
     * Get the front element.
     */
    public int peek() {
        transfer();
        return outStack.peek();
    }

    /**
     * Returns whether the queue is empty.
     */
    public boolean empty() {
        return outStack.empty() && inStack.empty();
    }
```

2.LeetCode 682 棒球比赛

题目链接

```java
public int calPoints(String[] ops) {
    Stack<Integer> stack = new Stack<>();
    for (String op : ops) {
        if (op.equals("+")) {
            int a = stack.pop();
            int b = stack.peek() + a;
            stack.push(a);
            stack.push(b);
        } else if (op.equals("D")) {
            stack.push(2 * stack.peek());
        } else if (op.equals("C")) {
            stack.pop();
        } else {
            stack.push(Integer.parseInt(op));
        }
    }
    int result = 0;
    for (Integer integer : stack) {
        result += integer;
    }
    return result;
}
```

3.LeetCode 844. 比较含退格的字符串

题目链接

```java
public boolean backspaceCompare(String S, String T) {
    return backspace(S).equals(backspace(T));
}

public String backspace(String str) {
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        char ch = str.charAt(i);
        if (ch == '#') {
            if (stringBuilder.length() > 0)
                stringBuilder.deleteCharAt(stringBuilder.length() - 1);
        } else {
            stringBuilder.append(ch);
        }
    }
    return stringBuilder.toString();
}
```

双指针法

```java
public static boolean backspaceCompare(String S, String T) {
    int countS = 0, countT = 0;
    int i = S.length() - 1, j = T.length() - 1;
    while (i >= 0 || j >= 0) {
        while (i >= 0) {
            if (S.charAt(i) == '#') {
                countS++;
                i--;
            } else if (countS > 0) {
                i--;
                countS--;
            } else {
                break;
            }
        }
        while (j >= 0) {
            if (T.charAt(j) == '#') {
                countT++;
                j--;
            } else if (countT > 0) {
                j--;
                countT--;
            } else {
                break;
            }
        }
        if (i >= 0 && j >= 0) {
            if (S.charAt(i) != T.charAt(j)) return false;
        } else {
            if (i >= 0 || j >= 0) return false;
        }
        i--;
        j--;
    }
    return true;
}
```

4.LeetCode 946 验证栈序列

题目链接

```java
    public boolean validateStackSequences(int[] pushed, int[] popped) {
        Stack<Integer> stack = new Stack<>();
        int j = 0;
        for (int n : pushed) {
            stack.push(n);
            while (!stack.isEmpty() && j < popped.length && stack.peek() ==
popped[j]) {
                stack.pop();
                j++;
            }
        }
        return stack.isEmpty();
    }
```

5.LeetCode 20 有效的括号

题目链接

```java
    public boolean isValid(String s) {
        HashMap<Character, Character> map = new HashMap<>();
        map.put(')', '(');
        map.put(']', '[');
        map.put('}', '{');
        Stack<Character> stack = new Stack<>();
        for (int i = 0; i < s.length(); i++) {
            switch (s.charAt(i)) {
                case '(':
                case '[':
                case '{':
                    stack.push(s.charAt(i));
                    break;
                case ')':
                case ']':
                case '}':
                    if (stack.isEmpty() || map.get(s.charAt(i)) != stack.peek())
return false;
                    stack.pop();
                    break;
            }
        }
        return stack.isEmpty();
    }
```

6.LeetCode 1021 删除最外层的括号

题目链接

```java
    public String removeOuterParentheses(String S) {
        StringBuilder stringBuilder = new StringBuilder();
        for (int i = 0, pre = 0, count = 0; i < S.length(); i++) {
            if (S.charAt(i) == '(') {
                count++;
            } else {
                count--;
            }
            if (count != 0) continue;
            stringBuilder.append(S.substring(pre + 1, i));
            pre = i + 1;
        }
        return stringBuilder.toString();
    }
```

进阶

```java
class Solution {
    public String removeOuterParentheses(String S) {
        StringBuilder sb = new StringBuilder();
        int level = 0;
        for (char c : S.toCharArray()) {
            if (c == ')') --level;
            if (level >= 1) sb.append(c);
            if (c == '(') ++level;
        }
        return sb.toString();
    }
}
```

7.LeetCode 1249. 移除无效的括号

题目链接

```
1
```

8.LeetCode 145. 二叉树的后序遍历

题目链接

递归法

```
1
```

迭代法

```
1
```

9.LeetCode 331 验证二叉树的前序序列化

题目链接

```
1
```

10.LeetCode 227 基本计算器II

题目链接

```
1
```

11.LeetCode 636 函数的独占时间

题目链接

```
1
```

12.LeetCode 1124. 表现良好的最长时间段

题目链接

kaikeba 开课吧