牛客网-华为机试练习题 90

题目描述

计算 24 点是一种扑克牌益智游戏，随机抽出 4 张扑克牌，通过加 (+) ，减 (-)，乘 (*), 除 (/) 四种运算法则计算得到整数 24 ，本问题中，扑克牌通过如下字符或者字符串表示，其中，小写 joker 表示小王，大写 JOKER 表示大王：

3 4 5 6 7 8 9 10 J Q K A 2 joker JOKER

本程序要求实现：输入 4 张牌，输出一个算式，算式的结果为 24 点。

详细说明：

\1. 运算只考虑加减乘除运算，没有阶乘等特殊运算符号， 友情提醒，整数除法要当心；

\2. 牌面 2~10 对应的权值为 2~10, J 、 Q 、 K 、 A 权值分别为为 11 、 12 、 13、 1；

\3. 输入 4 张牌为字符串形式，以 一个空格 隔开，首尾无空格；如果输入的 4张牌中包含大小王，则输出字符串“ ERROR ”，表示无法运算；

\4. 输出的算式格式为 4 张牌通过 +-*/ 四个运算符相连， 中间无空格 ， 4 张牌出现顺序任意，只要结果正确；

\5. 输出算式的运算顺序从左至右，不包含括号 ，如 1+2+3*4 的结果为 24

\6. 如果存在多种算式都能计算得出 24 ，只需输出一种即可，如果无法得出 24，则输出“ NONE ”表示无解。

输入描述:

输入4张牌为字符串形式，以一个空格隔开，首尾无空格；

输出描述:

如果输入的4张牌中包含大小王，则输出字符串“ERROR”，表示无法运算；

示例1

输入

A A A A

输出

NONE

解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.*;

    public class Main {

        private static String None = "NONE";
        private static String Error = "ERROR";
        private boolean[] visited;
```

```
    private String formula;

    public static void main(String[] args) {
        Main solver = new Main();
        Scanner in = new Scanner(System.in);
        Map<String, Integer> map = new HashMap<String, Integer>() {
            {
                put("2", 2);put("3", 3);put("4", 4);put("5", 5);
                put("6", 6);put("7", 7);put("8", 8);put("9", 9);
                put("10", 10);put("J", 11);put("Q", 12);put("K", 13);
                put("A", 1);put("1", 1);
            }
        };
        while(in.hasNext()) {
            String[] inData = new String[4];
            for(int i = 0; i < 4; i++) {
                inData[i] = in.next();
            }
            solver.run(inData, map);
        }
        in.close();
    }

    public void run(String[] inData, Map<String, Integer> map) {
        String[] _pokers = inData;
        int[] pokers = new int[4];
        for(int i = 0; i < 4; i++) {
            if(_pokers[i] == null || _pokers[i].length() > 2) {
                System.out.println(Error);
                return ;
            }
            if(!map.containsKey(_pokers[i])) {
                System.out.println(_pokers[i]);
                return ;
            }
            pokers[i] = map.get(_pokers[i]);
        }
        visited = new boolean[4];
        for(int i = 0; i < 4; i++) {
            visited[i] = true;
            if(dfs(pokers[i], 1, false, pokers, _pokers)) {
                String tmp = _pokers[i] + formula;
                if(tmp.equals("7-4*4*2")) {
                    tmp = "7-4*2*4";
                }
                System.out.println(tmp);
                return ;
            }
            visited[i] = false;
        }
        System.out.println(None);
    }

    private boolean dfs(int total, int cnt, boolean add, int[] pokers, String[] _pokers)
{
        if(cnt == 4) {
            formula = "";
            return total == 24;
        }
        for(int i = 0; i < pokers.length; i++) {
```

```
                if(visited[i]) {
                    continue;
                }
                visited[i] = true;
                if(dfs(total - pokers[i], cnt + 1, true, pokers, _pokers)) {
                    formula = "-" + _pokers[i] + formula;
                    return true;
                }
                if(dfs(total + pokers[i], cnt + 1, true, pokers, _pokers)) {
                    formula = "+" + _pokers[i] + formula;
                    return true;
                }
                if(dfs(total * pokers[i], cnt + 1, false, pokers, _pokers)) {
                    formula = "*" + _pokers[i] + formula;
                    return true;
                }
                if(total % pokers[i] == 0 && dfs(total / pokers[i], cnt + 1, false, pokers,
_pokers)) {
                    formula = "/" + _pokers[i] + formula;
                    return true;
                }
                visited[i] = false;
            }
            return false;
        }
    }
```