Java方向编程题答案

day22

[编程题]24534-到底买不买

链接：https://www.nowcoder.com/questionTerminal/2f13c507654b4f878b703cfbb5cdf3a5

【题目解析】

无

【解题思路】

该题目善用Java的集合框架就比较容易解决。

- 统计商人手上的珠子不同颜色的数量（HashMap）
- 统计用户手上的珠子不同颜色的数量（HashMap）
- 以用户为参考目标，判断商人是否存在用户要求的颜色和数量，进行统计差值和是否满足条件

【示例代码】

```
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Scanner;

public class Main {
public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);

  //商人输入
  String have = sc.nextLine();
  //用户输入
  String need = sc.nextLine();


  //商人手上每个珠子的数量统计
  Map<Character, Integer> h = new HashMap<>();
  for (char c : have.toCharArray()) {
      if (h.containsKey(c)) {
          h.put(c, h.get(c) + 1);
      } else {
          h.put(c, 1);
      }
  }

  //用户手上每个珠子的数量统计
  Map<Character, Integer> n = new HashMap<>();

  for (char c : need.toCharArray()) {
```

```
            if (n.containsKey(c)) {
                n.put(c, n.get(c) + 1);
            } else {
                n.put(c, 1);
            }
        }

        //计算差值，以用户为参考
        boolean weatherBy = true;
        int lack = 0;
        for (Entry<Character, Integer> en : n.entrySet()) {
            char k = en.getKey();
            int v = en.getValue();
            if (h.containsKey(k) && h.get(k) < v) {//商人的珠子包含用户的珠子但是不够
                weatherBy = false;
                lack += v - h.get(k);
            } else if (!h.containsKey(k)) {//商人的珠子不包含用户的珠子
                weatherBy = false;
                lack += v;
            }
        }
        if (weatherBy) {
            System.out.println("Yes " + (have.length() - need.length()));
        } else {
            System.out.println("No " + lack);
        }
    }
    }
```

[编程题]24914-链式A+B

链接：https://www.nowcoder.com/questionTerminal/ed85a09f0df047119e94fb3e5569855a

【题目解析】

无

【解题思路】

该题目的难度取决于解题思路。

- 第一种：采用链表遍历，节点元素求和加进位计算(较为复杂)
- 第二种：将链表转换为整数，进行求和计算，然后将整数又转换为链表即可（较为简单，此处选择第二种方式）

【示例代码】

```
public class Plus {

    public ListNode plusAB(ListNode a, ListNode b) {
        //讲链表转换为整数
        int aValue = listNodeConvertIntValue(a);
        int bValue = listNodeConvertIntValue(b);

        //计算求和
```

```
        int sumValue = aValue + bValue;
        //将整数转换为链表
        return intValueConvertListNode(sumValue);
    }

    private int listNodeConvertIntValue(ListNode node) {
        StringBuilder sb = new StringBuilder();
        ListNode curr = node;
        while (curr != null) {
            sb.append(curr.val);
            curr = curr.next;
        }
        return Integer.parseInt(sb.reverse().toString());
    }

    private ListNode intValueConvertListNode(int value) {
        char[] charArray = String.valueOf(value).toCharArray();
        ListNode node = new
ListNode(Integer.parseInt(String.valueOf(charArray[charArray.length - 1])));
        ListNode cur = node;
        //整数反转存储到链表中
        for (int i = charArray.length - 2; i >= 0; i--) {
            ListNode newNode = new ListNode(Integer.parseInt(String.valueOf(charArray[i])));
            cur.next = newNode;
            cur = newNode;
        }
        return node;
       }

    }

    class ListNode {
        int val;
        ListNode next = null;

        ListNode(int val) {
            this.val = val;
        }
    }
```