

Java-排序-盘的容量单位常用的有 M, G, T 这三个等级

题目描述:

磁盘的容量单位常用的有 M, G, T 这三个等级, 它们之间的换算关系为 $1T = 1024G$, $1G = 1024M$, 现在给定 n 块磁盘的容量, 请对它们按从小到大的顺序进行稳定排序, 例如给定 5 块盘的容量, 1T, 20M, 3G, 10G6T, 3M12G9M 排序后的结果为 20M, 3G, 3M12G9M, 1T, 10G6T。注意单位可以重复出现, 上述 3M12G9M 表示的容量即为 $3M+12G+9M$, 和 12M12G 相等。

输入描述:

输入第一行包含一个整数 $n(2 \leq n \leq 100)$, 表示磁盘的个数, 接下的 n 行, 每行一个字符串 (长度大于 2, 小于 30), 表示磁盘的容量, 由一个或多个格式为 mv 的子串组成, 其中 m 表示容量大小, v 表示容量单位, 例如 20M, 1T, 30G, 10G6T, 3M12G9M。

磁盘容量 m 的范围为 1 到 1024 的正整数, 容量单位 v 的范围只包含题目中提到的 M, G, T 三种, 换算关系如题目描述。

输出描述:

输出 n 行, 表示 n 块磁盘容量排序后的结果。

补充说明:

示例 1

输入:

```
3
1G
2G
1024M
```

输出:

```
1G
1024M
2G
```

说明:

1G 和 1024M 容量相等, 稳定排序要求保留它们原来的相对位置, 故 1G 在 1024M 之前

示例 2

输入:

```
3
2G4M
3M2G
1T
```

输出:

```
3M2G
2G4M
1T
```

说明:

1T 的容量大于 2G4M, 2G4M 的容量大于 3M2G

```
import java.util.*;
```

```
// 注意类名必须为 Main, 不要有任何 package xxx 信息
```

```

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        String temp = in.nextLine();
        int k = Integer.valueOf(temp);
        // System.out.println(k);
        List<String[]> list = new ArrayList<>();
        for(int i = 0 ; i < k; i++){
            list.add(helper(in.nextLine()));
        }
        Collections.sort(list, (o1,o2)->Integer.valueOf(o1[1]) - Integer.valueOf(o2[1]));
        for(int i = 0; i < k;i++){
            System.out.println(list.get(i)[0]);
        }
    }
    public static String[] helper(String s){
        // System.out.println(s);
        String[] res = new String[2];
        char[] c = s.toCharArray();
        int index = 0;
        Long temp = (long)0;
        while(index < c.length){
            int sum = 0;
            while(Character.isDigit(c[index])){
                sum = sum * 10 + (c[index] - '0');
                index++;
            }
            // System.out.println(sum);
            if(c[index] == 'T'){
                // res[1] = res[1] + String.valueOf(sum * 1024*1024);
                temp = temp + sum*1024*1024;
            }
            else if(c[index] == 'G'){
                // res[1] = res[1] + String.valueOf(sum * 1024);
                temp = temp + sum*1024;
            }
            else if(c[index] == 'M'){
                // res[1] = res[1] + String.valueOf(sum);
                temp = temp + sum;
            }
            index++;
        }
        res[0] = s;
    }
}

```

```
        res[1] = String.valueOf(temp);  
        // System.out.println(res[1]);  
        return res;  
    }  
  
}
```