

# 代码逻辑训练题

(第5次)

难度: 简单, 一般 限时: 60 分钟

#### 注:本次测试题为2题

- 文件编码必须为 UTF-8
- Java 只允许使用 JDK 中的 API,不允许使用第三方 jar; 包名必须按照要求命名
- JavaScript 只允许使用原生,不要引入第三方 js 文件,且最终以 js 文件方式提交,不要将结果写在 html 中。 如违反上述条例均以签名错误判定。

# 题目一

#### 问题描述

最近,实验室里以往高性能的电脑表现得越来越迟缓。于是你开始意识到,虽然电脑上有很多可用的磁盘空间,但是它已经被分割在了许多驱动器上,因此导致电脑越来越慢。你决定做些什么来改善电脑的性能,那就是尽量将所有数据合并到一起,并尽可能少的占据驱动器的数量。

给定一个参数 int[] used,表示每个驱动器上使用的磁盘空间的数量,再给定一个 int[] total,表示每个驱动器上磁盘的总容量,请尽可能将数合并到一起,并尽可能少的占用驱动器硬盘。

你可以假设驱动器上的数据是由非常非常小的文件组成,并且将文件从一个驱动器移动 到其他驱动器上不会出现问题。返回在尽可能合并数据后,占用驱动器的最小数量。

### 定义

#### Java

包名:	自己名字的缩写,如: package lhg;
类名:	DiskSpace
方法:	minDrives
参数:	int[],int[]
返回值:	int
方法签名:	public int minDrives(int[] used, int[] total)



# **JavaScript**

文件名:	DiskSpace.js
函数名:	minDrives
参数:	数字数组,数字数组
返回值:	数字
方法签名:	function minDrives (used, total)

## 约束

- 1. 参数 used 数组包含的元素数量只会在 1 到 50 之间 (包含 1 和 50),并且每一个元素的数值只会在 1-1000 之间 (包含 1 和 1000)
- 2. 参数 total 数组包含的元素数量和 used 相同,并且每一个元素的数值只会在 1-1000 之间 (包含 1 和 1000)
- 3. used 和 total 数组对应的元素 i, used[i]的数值总是小于等于 total[i]

## 示例

输入	返回
{300,525,110}	2
{350,600,115}	说明:在这个例子中,计算机包含三个驱动器,其中:
	驱动器 1: 总共 350 MB, 被使用 300 MB, 空闲 50 MB
	驱动器 2: 总共 600 MB, 被使用 525 MB, 空闲 75 MB
	驱动器 3: 总共 115 MB, 被使用 110 MB, 空闲 5 MB
	所以如果想占用最少的驱动器,那么就需要把"驱动器 3"上的
	50 MB 文件移动到"驱动器 1"上,将"驱动器 1"用满。之后再将
	"驱动器 3"上的剩余的 60 MB 文件移动到"驱动器 2"上,这样"
	驱动器 3"空闲,合并数据后只有两个驱动器被占用,所以方
	法返回 2。
{1,200,200,199,200,200}	1
{1000,200,200,200,200,200}	
{750,800,850,900,950}	5
{800,850,900,950,1000}	
{49,49,49,49,49,49,49,49,49,49,49,49,	49
49,49,49,49,49,49,49,49,49,49,49,4	
9,49,49,49,49,49,49,49,49,49,49	
,49,49,49,49,49,49,49,49,49,49,	
49}	
{50,50,50,50,50,50,50,50,50,50,50,50,	
50,50,50,50,50,50,50,50,50,50,50,50,5	
0,50,50,50,50,50,50,50,50,50,50,50	
,50,50,50,50,50,50,50,50,50,50,50,	
50}	



{331,242,384,366,428,114,145,89,381	6
,170,329,190,482,246,2,38,220,290,40	
2,385}	
{992,509,997,946,976,873,771,565,69	
3,714,755,878,897,789,969,727,765,5	
21,961,906}	

# 题目二

### 问题描述

某 24 小时便利店对它的员工采用时薪制,通常便利店的员工在进入和离开便利店的时候都会打卡,然后便利店老板会根据打卡记录计算员工的工作时间,因此如果给定一个员工的打卡记录和时薪,那么就可以计算该员工当日的工资。

员工进入和离开便利店的打卡时间会以"hh:mm:ss"的格式记录,其中 hh 表示小时(从 00 到 23),mm 表示分钟(从 00 到 59),ss 表示秒钟(从 00 到 59),时分秒均由两位 数字表示。员工的工作时间即为进入和离开便利店的打卡记录的间隔,比如某员工进入便利店的时间为 09:00:00,离开的时间为 17:30:00,则他的工作时间为 8 小时 30 分钟 0 秒。

另外如果员工工作在 **18**:00:00 到 **24**:00:00 之间或者工作在 **00**:00:00 到 **06**:00:00 之间,则这两个时段的时薪按照普通时薪的 **1**.5 倍支付。

请创建一个类 Salary,包含一个方法 howMuch,给定两个 String[], arrival 和 departure 作为参数,表示某一员工在一天内进入和离开便利店的一组打卡记录,再给定一个 int 参数 wage,表示该员工工作的普通时薪。请编写该方法返回员工在当日工作应得到的报酬。返回值向下取整,以 int 类型返回,比如如果员工当天的报酬为 15.865,则返回 15,去掉后面的小数部分。

### 定义

#### Java

包名:	自己名字的缩写,如: package lhg;
类名:	Salary
方法:	howMuch
参数:	String[],String[],int
返回值:	int
方法签名:	public int howMuch(String[] arrival, String[] departure, int wage)



## **JavaScript**

文件名:	Salary.js
函数名:	howMuch
参数:	字符串数组,字符串数组,数字
返回值:	数字
方法签名:	function howMuch (arrival, departure, wage)

## 约束

- 1. 参数 arrival 数组包含的元素数量只会在 1 到 10 之间(包含 1 和 10)
- 2. 参数 departure 包含的元素数量和 arrival 相同
- 3. arrival 和 departure 中元素的格式为"hh:mm:ss", 其中 hh 数值从 00 到 23, mm 数值从 00 到 59, ss 数值从 00 到 59
- 4. arrival 和 departure 中元素的时间将严格按照升序排列,就是说 arrival[0]<departure[0]<arrival[1]<departure[1]......,以此类推。
- 5. wage 的值为 100 到 10000, 包含 100 和 10000。

### 示例

输入	返回
{"08:00:00","13:00:00","19:27:32"}	10015
{"12:00:00","17:00:00","20:48:10"}	说明: 该员工第一段工作时间为 08:00:00 到
1000	12:00:00, 时间为 4 小时, 工资为 4000。第二段工
	作时间为 13:00:00 到 17:00:00,时间为 4 小时,工
	资为 4000。第三段工作时间为 19:27:32 到 20:48:10,
	时间为 1 小时 20 分 38 秒,工资为
	(1+20/60+38/3600)*1000*1.5 = 2015.83333,则该
	员 工 当 日 的 工 资 总 和 为
	4000+4000+2015.83333=10015.83333,向下取整,
	返回 10015。
{"01:05:47","16:48:12"}	33920
{"09:27:30","21:17:59"}	
2000	
{"00:00:00"}	299995
{"23:59:59"}	
10000	
{"10:00:00"}	80000
{"18:00:00"}	
10000	



{"22:19:46"}	7049
{"23:12:46"}	
5320	