代码逻辑训练题

（第 7 次）

难度：简单，中难　　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；Java 版本限定 1.6。**
- **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

一个简单、易于记忆的整数编码系统是非常有用的。通常可以使用一个替换代码，其中每个数字都可以使用一个字母进行编码。比如选取一个方便记忆的 10 个字母组成的短语作为编码的 key，短语的第一个字母可以由数字 1 代替，第二个字母可以由数字 2 代替，以此类推，最后一个字母可以由数字 0 代替。那么没有在短语中出现的字母可以忽略，并不影响解码的数值，这样编码便更不容易破译。

创建一个类 Substitute 包含方法 getValue，给定字符串 key 和 code，根据 key 计算 code 解码后的数值。

定义

Java

包名：	自己名字的缩写，如：package lhg;
类名：	Substitute
方法：	getValue
参数：	String,String
返回值：	int
方法签名：	public int getValue(String key, String code)

JavaScript

文件名：	Substitute.js
函数名：	getValue
参数：	字符串，字符串
返回值：	数字
方法签名：	function getValue (key, code)

约束

1. 参数 key 只会包含 10 个大写的‘A’-'Z'中的字符，并且 10 个字符没有重复
2. 参数 code 只会包含 1 到 9 个字符（包含 1 和 9），并且字符中至少有一个字符可以在 key 中找到。

示例

输入	返回
"TRADINGFEW" "LGXWEV"	709 说明：L,X 和 V 字母没有在"TRADINGFEW"中找到，可以被忽略。字母 G 在"TRADINGFEW"中是第 7 个字母, W 是最后一个字母, E 是第 9 个字母，所以按照上述编码规则，7 替换 G、0 替换 W、9 替换 E、其他字母忽略，因此"LGXWEV"解码后返回 709。
"ABCDEFGHIJ" "XJ"	0
"CRYSTALBUM" "MMA"	6

题目二

问题描述

你想加快你的程序运行速度，你打算通过编写多处理器并行运行任务来达到这个目的。你的应用程序执行 K 个独立任务，每个任务在一个处理器上运行 1 毫秒。在多个处理器上分配任务并不能使它运行得更快，但是在不同的处理器上运行不同的任务确实会使应用程序更快。

不幸的是，当应用程序在多个处理器上运行时，处理器之间的通信开销成为一个重要因素。尤其是每一对处理器之间都要首先花上一些毫秒进行通信，然后才能开始执行工作任务。更糟糕的是，由于处理器共享一个通信总线，不同配对的处理器之间不能并行通信。例如，如果处理器间的通信开销是 2 毫秒并且程序在 3 个处理器上运行，这将在实际运行任务前进行 6 毫秒的延迟用于处理器间的通信：即处理器 1 和 2 之间的通信使用 2 毫秒，处理器 1 和 3 通信使用 2 毫秒，和处理器 2 和 3 通信使用 2 毫秒，共 6 毫秒。注意，一旦初始通信阶段完成，即使每个处理器执行多个任务，也不需要进行进一步的通信。你的任务是确定在最少时间内运行 k 任务的处理器数量，假设每对处理器的通信开销超过毫秒。如果处理器的几种配置都是最小的时间，返回使用最少的处理器数量配置。

定义

Java

包名：	自己名字的缩写，如：package lhg;
类名：	ParallelSpeedup
方法：	numProcessors
参数：	int,int
返回值：	int
方法签名：	public int numProcessors(int k, int overhead)

JavaScript

文件名：	ParallelSpeedup.js
函数名：	numProcessors
参数：	数字,数字
返回值：	数字
方法签名：	function numProcessors (k, overhead)

限制

1. 执行方法的计算时间不能超过 2 秒

约束

1. k 的数值只会在 1 到 1000000 之间（包含 1 和 1000000）
2. overhead 的数值只会在 1 到 10 之间 （包含 1 和 10）

示例

输入	返回
12 1	2 说明：程序如果在两个处理器上执行将使用 7 毫秒 (1 毫秒用于每对处理器间的通信，6 毫秒用于任务执行，每个任务执行 1 毫秒). 虽然在 3 个处理器上执行也将使用 7 毫秒(3 毫秒用于每对处理器间的通信，4 毫秒用于任务执行). 但是我们需要返回处理器使用数量最少的方案，所以返回 2
50 3	3 说明：程序使用 3 个处理器运行 26 毫秒，其中 9 毫秒用于处理器间通信，17 毫秒用于执行程序（其中两个处理器执行 17 个任务，另一个处理器执行 16 个任务并等待其他处理器执行完任务，所以执行时间为 17 毫秒）
9 10	1
3333 2	12