**2024年校技能大赛Java程序设计赛项竞赛规程**

**一、竞赛项目名称**

 **Java程序设计大赛**

**二、竞赛目的**

通过本次大赛激发大学生的创新思维和分析问题、解决问题的能力，提高学生计算机的编程技术，从而营造良好的学科竞赛气氛。大赛为省技能大赛培养后备人才，形成可持续发展的梯队，促进学院人才培养质量的全面提升。

**三、组织机构**

（一）领导小组及职责

 主任：杨勇 韩永印

 副主任：龙浩 孟丽萍

 竞赛委员会工作职责：

 （1）负责竞赛工作例会，对例会议定的事进行督办；

 （2）负责竞赛场监考与裁判专家人选；

 （3）负责作品展示赛项作品评审

（二）竞赛工作小组及职责

 1、竞赛事务组

 组长：张敬斋

 成员：李梦梦、陆玉阳、张李彦（辅导员、实验员、学生志愿者）。

 工作职责：

 （1）赛场布置安排；（软件安装、设备维护）

 （2）颁奖仪式会务；（会标、会议议程、奖状、奖金）

 （3）资格审查、后勤保障、安全保卫。

 2、竞赛试题组

 组长：吕萍丽

 成员：王丽娟、张欣

 工作职责:

 （1）负责制定竞赛试题内容；

 （2）负责起草评判准则与标准。

 3、裁判组

 裁判长: 龙浩

 成员：吕萍丽、王丽娟等。

 工作职责：

 （1）根据竞赛组委会分工做好计分工作；

 （2）负责竞赛结果审议与仲裁，报竞赛委员会颁布。

 **四、参赛对象**

 信息工程学院学生。

  **五、竞赛方式及内容**

 （一）现场赛

 1.竞赛形式

个人赛（试题形式，一人一机，全程机考，选手现场参赛，规定时间内完成。）。

1. 个人赛一人一机。
2. 比赛采用客观题形式，考核Java语言的基础知识。选手指定指导教师进行辅导。
3. 比赛题型参照附件1。

 2.试题形式

比题型如下，根据选手所提交答案的测评结果为评分依据。

 2.1 代码填空题

题目描述一个具有确定解的问题。

题目同时给出该问题的某一解法的代码，但其中有缺失部分。

要求选手读懂代码逻辑，对其中的空缺部分补充代码，使整段代码完整。只填写空缺部分，不要填写完整句子。

不要写注释、说明或其它题目中未要求的内容。

 2.2. 编程大题

题目为若干具有一定难度梯度、分值不等的编程题目。

这些题目的要求明确、答案客观。

题目一般要用到标准输入和输出。要求选手通过编程，对给定的标准输入求解，并通过标准输出，按题目要求的格式输出解。题目一般会给出示例数据。

一般题目的难度主要集中于对算法的设计和逻辑的组织上。理论上，选手不可能通过猜测或其它非编程的手段获得问题的解。

选手给出的解法应具有普遍性，不能只适用于题目的示例数据（当然，至少应该适用于题目的示例数据）。

3. 参赛选手机器环境机器配置

X86兼容机器，内存不小于1G，硬盘不小于60G

操作系统Windows10、11

Java 语言开发环境：

 JDK 1.6

 Eclipse Helios for JavaSE

 API 帮助文档（中文，chm格式）

4. 试题涉及的基础知识

解题所涉及的知识：基本语法、面向对象、网络编程、接口、集合、IO、多线程、内部类、异常与保护，基本数据结构。

六、竞赛时间地点

现场个人赛比赛时间：6月14日下午15：00-16：30，比赛地点：B18机房。下午14:30签到。

  **七、报名时间、地点**

 1、比赛报名时间：6月11日至6月13日

 2、报名地点：B18-1026室

3、报名方式：以班级为单位，各个班学委收集班内报名的学生信息，电子稿（见附件2）和纸质稿交于B18-1026室。

  **九、竞赛须知**

1、参赛选手提前30 分钟到达比赛现场检录抽取赛位，比赛正式开始后，迟到选手不得入场进行比赛。

2.比赛平台软件由赛项组委会提供，参赛选手不得自带软件。

3.参赛选手须在确认竞赛任务和现场条件无误后开始比赛。

4.比赛过程中，选手休息、饮食或如厕时间均计算在比赛时间内。

5.比赛过程中，参赛选手须严格遵守赛场纪律，并接受裁判员的监督和警示；若因设备故障导致选手中断或终止比赛，由裁判长视具体情况做出裁决。

6.比赛过程中，参赛选手由于操作失误导致设备不能正常工作，或造成安全事故不能进行比赛的，将被中止比赛。

7.比赛过程中，各参赛选手限定在自己的工作区域内完成比赛任务。

8.若参赛选手欲提前结束比赛，应向裁判员举手示意，比赛终止时间由裁判员记录，参赛结束比赛后不得再进行任何操作。

9.比赛结束时，参赛选手应结束操作，并按照要求提交比赛结果，经裁判员确认后方可离开赛场。

**十、奖项设置**

1.个人奖项：所有参赛选手均参加个人奖项评比，个人按比赛项目设一等奖、二等奖和三等奖，获奖比例分别为5%、15%和20%，颁发荣誉证书；竞赛的学生根据竞赛的成绩，从高到低排列。课程赛成绩计入课程总成绩比例为40%。

2.优秀指导教师奖：根据学校相关文件学院设优秀指导教师奖若干名。

3.获奖结果将在比赛后一周内公布比赛成绩并颁发证书。

**十一、申诉与仲裁**

1．参赛队对不符合竞赛规定的设备、软件，有失公正的评判、奖励，以及对工作人员的违规行为等，均可提出申诉。

2．申诉应在竞赛结束后1小时内提出，超过时效不予受理。申诉时，应按照规定的程序由参赛选手向赛项仲裁递交书面申诉报告。

3．申诉人不得采取过激行为刁难、攻击工作人员，否则视为放弃申诉。

4．赛项设仲裁接受由选手提出的对裁判结果等方面问题的申诉。赛项仲裁在接到申诉后的2小时内组织复议，并及时反馈复议结果。仲裁结果为最终结果。

信息工程学院

2024年5月30日

附件1

**徐州工业职业技术学院**

**Java程序设计初赛样题**

【代码填空--java】

下面的代码定义了一个方法hasSameChar，用于判定一个给定的串中是否含有重复的字

符，比如“about”中，就没有重复的字符，而“telecom”，“aabaa”中都含有重复的字符，

其中“e”重复了2次，而“a”重复了4次，这些都算作有重复。

请根据方法的说明，分析给出的源程序，并填写划线部分缺失的代码。

注意，只填写缺少的，不要重复周围已经给出的内容，也不要填写任何说明性文字等。

public class A

{

 /\*

 判断串s中是否含有重复出现的字符

 如果有重复则返回true

 其它情况返回false

 判断的思路是：从左到右扫描每个字符

 对当前的字符，从右向左在s串中搜索它的出现位置，可以用 lastIndexOf 方法

 如果找到的位置与当前的位置不同，则必然存在该字符的重复现象，即可返回true

 其它情况返回false

 在特殊情况下，比如传入的是空指针，或者s为空串，或者只含有1个字符，都不可能含有

重复字符，

 因此，这些情况直接返回false

 \*/

 public static boolean hasSameChar(String s){

 if(s==null || s.length()<2) return false;

 for(int i=0; i<s.length(); i++){

 char c = s.charAt(i);

 int k = s.lastIndexOf(c);

 if(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) return true;

 }

 return false;

 }

 public static void main(String[] args){

 System.out.println(hasSameChar("a")); //false

 System.out.println(hasSameChar("abcdefg")); //false

 System.out.println(hasSameChar("abacdefag")); //true

 System.out.println(hasSameChar("abcdebfg")); //true

}

}

【编程大题】

用天平称重时，我们希望用尽可能少的砝码组合称出尽可能多的重量。

如果只有5个砝码，重量分别是1，3，9，27，81。则它们可以组合称出1到121之间

任意整数重量（砝码允许放在左右两个盘中）。

本题目要求编程实现：对用户给定的重量，给出砝码组合方案。

例如：

用户输入：

5

程序输出：

9-3-1

用户输入：

19

程序输出：

27-9+1

要求程序输出的组合总是大数在前小数在后。

可以假设用户的输入的数字符合范围1~121。

【解题思路提示】

把已知的砝码序列记为：x1, x2, x3, x4, x5, x6 (这里多加一个标准砝码，为解题叙

述方便)

对于任意给定的重量x，如果刚好等于xi 则问题解决。

否则一定会位于两个标准砝码重量的中间，不妨设为：xi < x < xj

令 a = x-xi, b = xj-x

则，x 要么可以表示为： xi + a, 要么可以表示为： xj-b

这样问题就归结为怎样表示出 a 或 b

另一思路：对于每个xi，可以乘以一个系数ki，再求和。

ki 的数值无外乎：-1 0 1

这样，因为标准砝码的数量是很少的，就可以多层循环组合ki来求解。

附件2

徐州工业职业技术学院

Java程序设计初赛报名表

|  |  |  |  |
| --- | --- | --- | --- |
| 班级 | 姓名 | 学号 | 联系方式 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |