# 实验四 CSP认证2017年9月测试真题

【开发语言及实现平台或实验环境】

Windows2000 或XP，JDK1.6与Jcreator4.0

【实验目的】

1. 理解Java语言是如何体现面向对象编程基本思想，
2. 了解类的封装方法，以及如何创建类和对象，
3. 了解成员变量和成员方法的特性。
4. 了解类的继承性和多态性的作用。

【实验要求】

1. 理解Java语言是如何体现面向对象编程基本思想，
2. 了解类的封装方法，以及如何创建类和对象，
3. 了解成员变量和成员方法的特性。
4. 了解类的继承性和多态性的作用。
5. 编写体现类的继承性（成员变量，成员方法，成员变量隐藏）的程序。
6. 编写体现类多态性（成员方法重载，构造方法重载）的程序。
7. **在www.cspro.org中注册用户，在历年真题中提交验证，在实验报告中提交成绩截图。**

【注意事项】

1. 程序**没有**使用package语句来定义包的信息。（如果定义了将无法评测）
2. 程序的主类名称是**Main**，主类的定义是public class Main。我的程序运行的入口是主类Main中的main函数，定义为public static void main(String[] args)。
3. 程序是从**标准输入**(System.in)中读入数据的（使用Scanner等类来处理System.in的输入视为满足条件），结果是输出到**标准输出**(System.out)的。
4. 程序中**没有**“请输入n”之类的输入输出提示，也**没有**输出中间的结果，所有的输出内容都与题设中的输出格式相对应。我的输出格式（包括换行和大小写等）与题设中输出格式的要求相符。

【实验内容】**(五题之中任选三题，可根据自己能力具体选择)**

**一、打酱油**

|  |  |
| --- | --- |
| 试题编号： | 201709-1 |
| 试题名称： | 打酱油 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　小明带着*N*元钱去买酱油。酱油10块钱一瓶，商家进行促销，每买3瓶送1瓶，或者每买5瓶送2瓶。请问小明最多可以得到多少瓶酱油。  **输入格式**  　　输入的第一行包含一个整数*N*，表示小明可用于买酱油的钱数。*N*是10的整数倍，*N*不超过300。  **输出格式**  　　输出一个整数，表示小明最多可以得到多少瓶酱油。  **样例输入**  40  **样例输出**  5  **样例说明**  　　把40元分成30元和10元，分别买3瓶和1瓶，其中3瓶送1瓶，共得到5瓶。  **样例输入**  80  **样例输出**  11  **样例说明**  　　把80元分成30元和50元，分别买3瓶和5瓶，其中3瓶送1瓶，5瓶送2瓶，共得到11瓶。 |

**二、公共钥匙盒**

|  |  |
| --- | --- |
| 试题编号： | 201709-2 |
| 试题名称： | 公共钥匙盒 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　有一个学校的老师共用*N*个教室，按照规定，所有的钥匙都必须放在公共钥匙盒里，老师不能带钥匙回家。每次老师上课前，都从公共钥匙盒里找到自己上课的教室的钥匙去开门，上完课后，再将钥匙放回到钥匙盒中。 　　钥匙盒一共有*N*个挂钩，从左到右排成一排，用来挂*N*个教室的钥匙。一串钥匙没有固定的悬挂位置，但钥匙上有标识，所以老师们不会弄混钥匙。 　　每次取钥匙的时候，老师们都会找到自己所需要的钥匙将其取走，而不会移动其他钥匙。每次还钥匙的时候，还钥匙的老师会找到最左边的空的挂钩，将钥匙挂在这个挂钩上。如果有多位老师还钥匙，则他们按钥匙编号从小到大的顺序还。如果同一时刻既有老师还钥匙又有老师取钥匙，则老师们会先将钥匙全还回去再取出。 　　今天开始的时候钥匙是按编号从小到大的顺序放在钥匙盒里的。有*K*位老师要上课，给出每位老师所需要的钥匙、开始上课的时间和上课的时长，假设下课时间就是还钥匙时间，请问最终钥匙盒里面钥匙的顺序是怎样的？  **输入格式**  　　输入的第一行包含两个整数*N*, *K*。 　　接下来*K*行，每行三个整数*w*, *s*, *c*，分别表示一位老师要使用的钥匙编号、开始上课的时间和上课的时长。可能有多位老师使用同一把钥匙，但是老师使用钥匙的时间不会重叠。 　　保证输入数据满足输入格式，你不用检查数据合法性。  **输出格式**  　　输出一行，包含*N*个整数，相邻整数间用一个空格分隔，依次表示每个挂钩上挂的钥匙编号。  **样例输入**  5 2 4 3 3 2 2 7  **样例输出**  1 4 3 2 5  **样例说明**  　　第一位老师从时刻3开始使用4号教室的钥匙，使用3单位时间，所以在时刻6还钥匙。第二位老师从时刻2开始使用钥匙，使用7单位时间，所以在时刻9还钥匙。 　　每个关键时刻后的钥匙状态如下（X表示空）： 　　时刻2后为1X345； 　　时刻3后为1X3X5； 　　时刻6后为143X5； 　　时刻9后为14325。  **样例输入**  5 7 1 1 14 3 3 12 1 15 12 2 7 20 3 18 12 4 21 19 5 30 9  **样例输出**  1 2 3 5 4  **评测用例规模与约定**  　　对于30%的评测用例，1 ≤ *N*, *K* ≤ 10, 1 ≤ *w* ≤ *N*, 1 ≤ *s*, *c* ≤ 30； 　　对于60%的评测用例，1 ≤ *N*, *K* ≤ 50，1 ≤ *w* ≤ *N*，1 ≤ *s* ≤ 300，1 ≤ *c* ≤ 50； 　　对于所有评测用例，1 ≤ *N*, *K* ≤ 1000，1 ≤ *w* ≤ *N*，1 ≤ *s* ≤ 10000，1 ≤ *c* ≤ 100。 |

**三、JSON查询**

|  |  |
| --- | --- |
| 试题编号： | 201709-3 |
| 试题名称： | JSON查询 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式，可以用来描述半结构化的数据。JSON 格式中的基本单元是值 (value)，出于简化的目的本题只涉及 2 种类型的值： 　　\* 字符串 (string)：字符串是由双引号 " 括起来的一组字符（可以为空）。如果字符串的内容中出现双引号 "，在双引号前面加反斜杠，也就是用 \" 表示；如果出现反斜杠 \，则用两个反斜杠 \\ 表示。反斜杠后面不能出现 " 和 \ 以外的字符。例如：""、"hello"、"\"\\"。 　　\* 对象 (object)：对象是一组键值对的无序集合（可以为空）。键值对表示对象的属性，键是属性名，值是属性的内容。对象以左花括号 { 开始，右花括号 } 结束，键值对之间以逗号 , 分隔。一个键值对的键和值之间以冒号 : 分隔。键必须是字符串，同一个对象所有键值对的键必须两两都不相同；值可以是字符串，也可以是另一个对象。例如：{}、{"foo": "bar"}、{"Mon": "weekday", "Tue": "weekday", "Sun": "weekend"}。 　　除了字符串内部的位置，其他位置都可以插入一个或多个空格使得 JSON 的呈现更加美观，也可以在一些地方换行，不会影响所表示的数据内容。例如，上面举例的最后一个 JSON 数据也可以写成如下形式。 　　{ 　　"Mon": "weekday", 　　"Tue": "weekday", 　　"Sun": "weekend" 　　} 　　给出一个 JSON 格式描述的数据，以及若干查询，编程返回这些查询的结果。  **输入格式**  　　第一行是两个正整数 *n* 和 *m*，分别表示 JSON 数据的行数和查询的个数。 　　接下来 *n* 行，描述一个 JSON 数据，保证输入是一个合法的 JSON 对象。 　　接下来 *m* 行，每行描述一个查询。给出要查询的属性名，要求返回对应属性的内容。需要支持多层查询，各层的属性名之间用小数点 . 连接。保证查询的格式都是合法的。  **输出格式**  　　对于输入的每一个查询，按顺序输出查询结果，每个结果占一行。 　　如果查询结果是一个字符串，则输出 STRING <string>，其中 <string> 是字符串的值，中间用一个空格分隔。 　　如果查询结果是一个对象，则输出 OBJECT，不需要输出对象的内容。 　　如果查询结果不存在，则输出 NOTEXIST。  **样例输入**  10 5 { "firstName": "John", "lastName": "Smith", "address": { "streetAddress": "2ndStreet", "city": "NewYork", "state": "NY" }, "esc\\aped": "\"hello\"" } firstName address address.city address.postal esc\aped  **样例输出**  STRING John OBJECT STRING NewYork NOTEXIST STRING "hello"  **评测用例规模与约定**  *n* ≤ 100，每行不超过 80 个字符。 　　*m* ≤ 100，每个查询的长度不超过 80 个字符。 　　字符串中的字符均为 ASCII 码 33-126 的可打印字符，不会出现空格。所有字符串都不是空串。 　　所有作为键的字符串不会包含小数点 .。查询时键的大小写敏感。 　　50%的评测用例输入的对象只有 1 层结构，80%的评测用例输入的对象结构层数不超过 2 层。举例来说，{"a": "b"} 是一层结构的对象，{"a": {"b": "c"}} 是二层结构的对象，以此类推。 |

**四、通信网络**

|  |  |
| --- | --- |
| 试题编号： | 201709-4 |
| 试题名称： | 通信网络 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　某国的军队由*N*个部门组成，为了提高安全性，部门之间建立了*M*条通路，每条通路只能单向传递信息，即一条从部门*a*到部门*b*的通路只能由*a*向*b*传递信息。信息可以通过中转的方式进行传递，即如果*a*能将信息传递到*b*，*b*又能将信息传递到*c*，则*a*能将信息传递到*c*。一条信息可能通过多次中转最终到达目的地。 　　由于保密工作做得很好，并不是所有部门之间都互相知道彼此的存在。只有当两个部门之间可以直接或间接传递信息时，他们才彼此知道对方的存在。部门之间不会把自己知道哪些部门告诉其他部门。 http://118.190.20.162/RequireFile.do?fid=yHg9gf9q 　　上图中给了一个4个部门的例子，图中的单向边表示通路。部门1可以将消息发送给所有部门，部门4可以接收所有部门的消息，所以部门1和部门4知道所有其他部门的存在。部门2和部门3之间没有任何方式可以发送消息，所以部门2和部门3互相不知道彼此的存在。 　　现在请问，有多少个部门知道所有*N*个部门的存在。或者说，有多少个部门所知道的部门数量（包括自己）正好是*N*。  **输入格式**  　　输入的第一行包含两个整数*N*, *M*，分别表示部门的数量和单向通路的数量。所有部门从1到*N*标号。 　　接下来*M*行，每行两个整数*a*, *b*，表示部门*a*到部门*b*有一条单向通路。  **输出格式**  　　输出一行，包含一个整数，表示答案。  **样例输入**  4 4 1 2 1 3 2 4 3 4  **样例输出**  2  **样例说明**  　　部门1和部门4知道所有其他部门的存在。  **评测用例规模与约定**  　　对于30%的评测用例，1 ≤ *N* ≤ 10，1 ≤ *M* ≤ 20； 　　对于60%的评测用例，1 ≤ *N* ≤ 100，1 ≤ *M* ≤ 1000； 　　对于100%的评测用例，1 ≤ *N* ≤ 1000，1 ≤ *M* ≤ 10000。 |

**五、除法**

|  |  |
| --- | --- |
| 试题编号： | 201709-5 |
| 试题名称： | 除法 |
| 时间限制： | 10.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　小葱喜欢除法，所以他给了你*N*个数*a*1, *a*2, ⋯, *aN*，并且希望你执行*M*次操作，每次操作可能有以下两种： 　　给你三个数*l*, *r*, *v*，你需要将*al*, *al*+1, ⋯, *ar*之间所有*v*的倍数除以*v*。 　　给你两个数*l*, *r*，你需要回答*al* + *al*+1+ ⋯ + *ar*的值是多少。  **输入格式**  　　第一行两个整数*N*, *M*，代表数的个数和操作的次数。 　　接下来一行*N*个整数，代表*N*个数一开始的值。 　　接下来*M*行，每行代表依次操作。每行开始有一个整数*opt*。如果*opt*=1，那么接下来有三个数*l*, *r*, *v*，代表这次操作需要将第*l*个数到第*r*个数中*v*的倍数除以*v*；如果*opt* = 2，那么接下来有两个数*l*, *r*，代表你需要回答第*l*个数到第*r*个数的和。  **输出格式**  　　对于每一次的第二种操作，输出一行代表这次操作所询问的值。  **样例输入**  5 3 1 2 3 4 5 2 1 5 1 1 3 2 2 1 5  **样例输出**  15 14  **评测用例规模与约定**  　　对于30%的评测用例，1 ≤ *N*, *M* ≤ 1000； 　　对于另外20%的评测用例，第一种操作中一定有*l* = *r*； 　　对于另外20%的评测用例，第一种操作中一定有*l* = 1 , *r* = *N*； 　　对于100%的评测用例，1 ≤ *N*, *M* ≤ 105，0 ≤ *a*1, *a*2, ⋯, *aN* ≤ 106, 1 ≤ *v*≤ 106, 1 ≤ *l* ≤ *r* ≤ *N*。 |