# 实验五 CSP认证2017年3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、分蛋糕**

|  |  |
| --- | --- |
| 试题编号： | 201703-1 |
| 试题名称： | 分蛋糕 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　小明今天生日，他有*n*块蛋糕要分给朋友们吃，这*n*块蛋糕（编号为1到*n*）的重量分别为*a*1, *a*2, …, *an*。小明想分给每个朋友至少重量为*k*的蛋糕。小明的朋友们已经排好队准备领蛋糕，对于每个朋友，小明总是先将自己手中编号最小的蛋糕分给他，当这个朋友所分得蛋糕的重量不到*k*时，再继续将剩下的蛋糕中编号最小的给他，直到小明的蛋糕分完或者这个朋友分到的蛋糕的总重量大于等于*k*。 　　请问当小明的蛋糕分完时，总共有多少个朋友分到了蛋糕。  **输入格式**  　　输入的第一行包含了两个整数*n*, *k*，意义如上所述。 　　第二行包含*n*个正整数，依次表示*a*1, *a*2, …, *an*。  **输出格式**  　　输出一个整数，表示有多少个朋友分到了蛋糕。  **样例输入**  6 9 2 6 5 6 3 5  **样例输出**  3  **样例说明**  　　第一个朋友分到了前3块蛋糕，第二个朋友分到了第4、5块蛋糕，第三个朋友分到了最后一块蛋糕。  **评测用例规模与约定**  　　对于所有评测用例，1 ≤ *n* ≤ 1000，1 ≤ *k* ≤ 10000，1 ≤ *ai* ≤ 1000。 |

**二、学生排队**

|  |  |
| --- | --- |
| 试题编号： | 201703-2 |
| 试题名称： | 学生排队 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　体育老师小明要将自己班上的学生按顺序排队。他首先让学生按学号从小到大的顺序排成一排，学号小的排在前面，然后进行多次调整。一次调整小明可能让一位同学出队，向前或者向后移动一段距离后再插入队列。 　　例如，下面给出了一组移动的例子，例子中学生的人数为8人。 　　0）初始队列中学生的学号依次为1, 2, 3, 4, 5, 6, 7, 8； 　　1）第一次调整，命令为“3号同学向后移动2”，表示3号同学出队，向后移动2名同学的距离，再插入到队列中，新队列中学生的学号依次为1, 2, 4, 5, 3, 6, 7, 8； 　　2）第二次调整，命令为“8号同学向前移动3”，表示8号同学出队，向前移动3名同学的距离，再插入到队列中，新队列中学生的学号依次为1, 2, 4, 5, 8, 3, 6, 7； 　　3）第三次调整，命令为“3号同学向前移动2”，表示3号同学出队，向前移动2名同学的距离，再插入到队列中，新队列中学生的学号依次为1, 2, 4, 3, 5, 8, 6, 7。 　　小明记录了所有调整的过程，请问，最终从前向后所有学生的学号依次是多少？ 　　请特别注意，上述移动过程中所涉及的号码指的是学号，而不是在队伍中的位置。在向后移动时，移动的距离不超过对应同学后面的人数，如果向后移动的距离正好等于对应同学后面的人数则该同学会移动到队列的最后面。在向前移动时，移动的距离不超过对应同学前面的人数，如果向前移动的距离正好等于对应同学前面的人数则该同学会移动到队列的最前面。  **输入格式**  　　输入的第一行包含一个整数*n*，表示学生的数量，学生的学号由1到*n*编号。 　　第二行包含一个整数*m*，表示调整的次数。 　　接下来m行，每行两个整数p, q，如果q为正，表示学号为p的同学向后移动q，如果q为负，表示学号为p的同学向前移动-q。  **输出格式**  　　输出一行，包含n个整数，相邻两个整数之间由一个空格分隔，表示最终从前向后所有学生的学号。  **样例输入**  8 3 3 2 8 -3 3 -2  **样例输出**  1 2 4 3 5 8 6 7  **评测用例规模与约定**  　　对于所有评测用例，1 ≤ *n* ≤ 1000，1 ≤ *m* ≤ 1000，所有移动均合法。 |

**三、Markdown**

|  |  |
| --- | --- |
| 试题编号： | 201703-3 |
| 试题名称： | Markdown |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　Markdown 是一种很流行的轻量级标记语言（lightweight markup language），广泛用于撰写带格式的文档。例如以下这段文本就是用 Markdown 的语法写成的：    　　这些用 Markdown 写成的文本，尽管本身是纯文本格式，然而读者可以很容易地看出它的文档结构。同时，还有很多工具可以自动把 Markdown 文本转换成 HTML 甚至 Word、PDF 等格式，取得更好的排版效果。例如上面这段文本通过转化得到的 HTML 代码如下所示：    　　本题要求由你来编写一个 Markdown 的转换工具，完成 Markdown 文本到 HTML 代码的转换工作。简化起见，本题定义的 Markdown 语法规则和转换规则描述如下： 　　●**区块**：区块是文档的顶级结构。本题的 Markdown 语法有 3 种区块格式。在输入中，相邻两个区块之间用一个或多个空行分隔。**输出时删除所有分隔区块的空行。** 　　○段落：一般情况下，连续多行输入构成一个段落。段落的转换规则是在段落的第一行行首插入 `<p>`，在最后一行行末插入 `</p>`。 　　○标题：每个标题区块只有一行，由若干个 `#` 开头，接着一个或多个空格，然后是标题内容，直到行末。`#` 的个数决定了标题的等级。转换时，`# Heading` 转换为 `<h1>Heading</h1>`，`## Heading` 转换为 `<h2>Heading</h2>`，以此类推。标题等级最深为 6。 　　○无序列表：无序列表由若干行组成，每行由 `\*` 开头，接着一个或多个空格，然后是列表项目的文字，直到行末。转换时，在最开始插入一行 `<ul>`，最后插入一行 `</ul>`；对于每行，`\* Item` 转换为 `<li>Item</li>`。本题中的无序列表只有一层，不会出现缩进的情况。 　　●行内：对于区块中的内容，有以下两种行内结构。 　　○强调：`\_Text\_` 转换为 `<em>Text</em>`。强调不会出现嵌套，每行中 `\_` 的个数一定是偶数，且不会连续相邻。注意 `\_Text\_` 的前后不一定是空格字符。 　　○超级链接：`[Text](Link)` 转换为 `<a href="Link">Text</a>`。超级链接和强调可以相互嵌套，但每种格式不会超过一层。  **输入格式**  　　输入由若干行组成，表示一个用本题规定的 Markdown 语法撰写的文档。  **输出格式**  　　输出由若干行组成，表示输入的 Markdown 文档转换成产生的 HTML 代码。  **样例输入**  # Hello  Hello, world!  **样例输出**  <h1>Hello</h1> <p>Hello, world!</p>  **评测用例规模与约定**  　　本题的测试点满足以下条件： 　　●本题每个测试点的输入数据所包含的行数都不超过100，每行字符的个数（包括行末换行符）都不超过100。 　　●除了换行符之外，所有字符都是 ASCII 码 32 至 126 的可打印字符。 　　●每行行首和行末都不会出现空格字符。 　　●输入数据除了 Markdown 语法所需，内容中不会出现 `#`、`\*`、`\_`、`[`、`]`、`(`、`)`、`<`、`>`、`&` 这些字符。 　　●所有测试点均符合题目所规定的 Markdown 语法，你的程序不需要考虑语法错误的情况。 　　每个测试点包含的语法规则如下表所示，其中“√”表示包含，“×”表示不包含。   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 测试点编号 | 段落 | 标题 | 无序列表 | 强调 | 超级链接 | | 1 | √ | × | × | × | × | | 2 | √ | √ | × | × | × | | 3 | √ | × | √ | × | × | | 4 | √ | × | × | √ | × | | 5 | √ | × | × | × | √ | | 6 | √ | √ | √ | × | × | | 7 | √ | × | × | √ | √ | | 8 | √ | √ | × | √ | × | | 9 | √ | × | √ | × | √ | | 10 | √ | √ | √ | √ | √ |   **提示**  　　由于本题要将输入数据当做一个文本文件来处理，要逐行读取直到文件结束，C/C++、Java 语言的用户可以参考以下代码片段来读取输入内容。 |

**四、地铁修建**

|  |  |
| --- | --- |
| 试题编号： | 201703-4 |
| 试题名称： | 地铁修建 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　A市有n个交通枢纽，其中1号和n号非常重要，为了加强运输能力，A市决定在1号到n号枢纽间修建一条地铁。 　　地铁由很多段隧道组成，每段隧道连接两个交通枢纽。经过勘探，有m段隧道作为候选，两个交通枢纽之间最多只有一条候选的隧道，没有隧道两端连接着同一个交通枢纽。 　　现在有n家隧道施工的公司，每段候选的隧道只能由一个公司施工，每家公司施工需要的天数一致。而每家公司最多只能修建一条候选隧道。所有公司同时开始施工。 　　作为项目负责人，你获得了候选隧道的信息，现在你可以按自己的想法选择一部分隧道进行施工，请问修建整条地铁最少需要多少天。  **输入格式**  　　输入的第一行包含两个整数*n*, *m*，用一个空格分隔，分别表示交通枢纽的数量和候选隧道的数量。 　　第2行到第*m*+1行，每行包含三个整数*a*, *b*, *c*，表示枢纽*a*和枢纽*b*之间可以修建一条隧道，需要的时间为*c*天。  **输出格式**  　　输出一个整数，修建整条地铁线路最少需要的天数。  **样例输入**  6 6 1 2 4 2 3 4 3 6 7 1 4 2 4 5 5 5 6 6  **样例输出**  6  **样例说明**  　　可以修建的线路有两种。 　　第一种经过的枢纽依次为1, 2, 3, 6，所需要的时间分别是4, 4, 7，则整条地铁线需要7天修完； 　　第二种经过的枢纽依次为1, 4, 5, 6，所需要的时间分别是2, 5, 6，则整条地铁线需要6天修完。 　　第二种方案所用的天数更少。  **评测用例规模与约定**  　　对于20%的评测用例，1 ≤ *n* ≤ 10，1 ≤ *m* ≤ 20； 　　对于40%的评测用例，1 ≤ *n* ≤ 100，1 ≤ *m* ≤ 1000； 　　对于60%的评测用例，1 ≤ *n* ≤ 1000，1 ≤ *m* ≤ 10000，1 ≤ *c* ≤ 1000； 　　对于80%的评测用例，1 ≤ *n* ≤ 10000，1 ≤ *m* ≤ 100000； 　　对于100%的评测用例，1 ≤ *n* ≤ 100000，1 ≤ *m* ≤ 200000，1 ≤ *a*, *b* ≤ *n*，1 ≤ *c* ≤ 1000000。  　　所有评测用例保证在所有候选隧道都修通时1号枢纽可以通过隧道到达其他所有枢纽。 |

**五、引水入城**

|  |  |
| --- | --- |
| 试题编号： | 201703-5 |
| 试题名称： | 引水入城 |
| 时间限制： | 2.0s |
| 内存限制： | 512.0MB |
| 问题描述： | **问题描述**  　　MF城建立在一片高原上。由于城市唯一的水源是位于河谷地带的湖中，人们在坡地上修筑了一片网格状的抽水水管，以将湖水抽入城市。如下图所示：   　　这片管网由 *n* 行 *m* 列节点（红色，图中 *n* = 5，*m* = 6），横向管道（紫色）和纵向管道（橙色）构成。 　　行和列分别用 1 到 *n* 的整数和 1 到 *m* 的整数表示。第 1 行的任何一个节点均可以抽取湖水，湖水到达第 *n* 行的任何一个节点即算作引入了城市。 　　除第一行和最后一行外，横向相邻或纵向相邻的两个节点之间一定有一段管道，每一段管道都有各自的最大的抽水速率，并需要根据情况选择抽水还是放水。对于纵向的管道（橙色），允许从上方向下方抽水或从下方向上方放水；如果从图中的上方向下方抽水，那么单位时间内能通过的水量不能超过管道的最大速率；如果从下方向上方放水，因为下方海拔较高，因此可以允许有任意大的水量。对于横向的管道（紫色），允许从左向右或从右向左抽水，不允许放水，两种情况下单位时间流过的水量都不能超过管道的最大速率。 　　现在MF城市的水务负责人想知道，在已知每个管道单位时间容量的情况下，MF城每单位时间最多可以引入多少的湖水。  **输入格式**  　　由于输入规模较大，我们采用伪随机生成的方式生成数据。 　　每组数据仅一行包含 6 个非负整数 *n*, *m*, *A*, *B*, *Q*, *X*0。其中，*n* 和 *m* 如前文所述，表示管网的大小，保证 2 ≤*n*, *m* ≤ 5000；保证 1 ≤ *A*, *B*, *Q*, *X*0 ≤ 109。 　　*A*, *B*, *Q*, *X*0 是数据生成的参数，我们用如下的方式定义一个数列 { *Xi* }： 　　*Xi*+1 = ( *AXi* + *B*) **mod** *Q*, (*i* ≥ 0) 　　我们将数列的第 1 项到第 (*n*-1)*m* 项作为纵向管道的单位时间容量，其中 *X*(*s*-1)*m*+*t* 表示第 *s* 行第 *t* 列的节点到第 *s*+1 行第 *t* 列管道单位时间的容量；将数列的第 (*n*-1)*m*+1 项到第 (*n*-1)*m*+**(*n*-2)(*m*-1)** 项（即接下来的 (*n*-2)(*m*-1) 项）作为横向管道的单位时间容量，其中 *X*(*n*-1)*m*+**(*s*-2)(*m*-1)**+*t* 表示第 *s* 行第 *t* 列的节点到第 *s* 行第 *t*+1 列管道单位时间的容量。  **输出格式**  　　输出一行一个整数，表示MF城每单位时间可以引入的水量。 　　注意计算过程中有些参数可能超过32位整型表示的最大值，请注意使用64位整型存储相应数据。  **样例输入**  3 3 10 3 19 7  **样例输出**  38  **样例说明**  　　使用参数得到数列 { *Xi* }={ 7, 16, 11, 18, 12, 9, 17, 2, 4, … }，按照输入格式可以得到每个管道的最大抽水量如下图所示：   　　在标准答案中，单位时间可以引水 38 单位。所有纵向管道均向下抽水即可，不需要横向管道抽水，也不需要向上放水。  **样例输入**  2 5 595829232 749238243 603779819 532737791  **样例输出**  1029036148  **样例输入**  5 2 634932890 335818535 550589587 977780683  **样例输出**  192923706  **样例输入**  5 5 695192542 779962396 647834146 157661239  **样例输出**  1449991168  **评测用例规模与约定**  　　共有10组评测数据，每组数据的参数规模如下所示：   |  |  |  | | --- | --- | --- | | 测试点编号 | *n* | *m* | | 1 | =2 | =1000 | | 2 | =1000 | =2 | | 3 | =1000 | =2 | | 4 | =5 | =5 | | 5 | =10 | =10 | | 6 | =100 | =100 | | 7 | =500 | =500 | | 8 | =1000 | =1000 | | 9 | =2000 | =2000 | | 10 | =5000 | =5000 | |