# 实验五 CSP认证2018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、跳一跳**

|  |  |
| --- | --- |
| 试题编号： | 201803-1 |
| 试题名称： | 跳一跳 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　近来，跳一跳这款小游戏风靡全国，受到不少玩家的喜爱。 　　简化后的跳一跳规则如下：玩家每次从当前方块跳到下一个方块，如果没有跳到下一个方块上则游戏结束。 　　如果跳到了方块上，但没有跳到方块的中心则获得1分；跳到方块中心时，若上一次的得分为1分或这是本局游戏的第一次跳跃则此次得分为2分，否则此次得分比上一次得分多两分（即连续跳到方块中心时，总得分将+2，+4，+6，+8...）。 　　现在给出一个人跳一跳的全过程，请你求出他本局游戏的得分（按照题目描述的规则）。  **输入格式**  　　输入包含多个数字，用空格分隔，每个数字都是1，2，0之一，1表示此次跳跃跳到了方块上但是没有跳到中心，2表示此次跳跃跳到了方块上并且跳到了方块中心，0表示此次跳跃没有跳到方块上（此时游戏结束）。  **输出格式**  　　输出一个整数，为本局游戏的得分（在本题的规则下）。  **样例输入**  1 1 2 2 2 1 1 2 2 0  **样例输出**  22  **数据规模和约定**  　　对于所有评测用例，输入的数字不超过30个，保证0正好出现一次且为最后一个数字。 |

**二、碰撞的小球**

|  |  |
| --- | --- |
| 试题编号： | 201803-2 |
| 试题名称： | 碰撞的小球 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　数轴上有一条长度为L（L为偶数)的线段，左端点在原点，右端点在坐标L处。有n个不计体积的小球在线段上，开始时所有的小球都处在偶数坐标上，速度方向向右，速度大小为1单位长度每秒。 　　当小球到达线段的端点（左端点或右端点）的时候，会立即向相反的方向移动，速度大小仍然为原来大小。 　　当两个小球撞到一起的时候，两个小球会分别向与自己原来移动的方向相反的方向，以原来的速度大小继续移动。 　　现在，告诉你线段的长度L，小球数量n，以及n个小球的初始位置，请你计算t秒之后，各个小球的位置。  **提示**  　　因为所有小球的初始位置都为偶数，而且线段的长度为偶数，可以证明，不会有三个小球同时相撞，小球到达线段端点以及小球之间的碰撞时刻均为整数。 　　同时也可以证明两个小球发生碰撞的位置一定是整数（但不一定是偶数）。  **输入格式**  　　输入的第一行包含三个整数n, L, t，用空格分隔，分别表示小球的个数、线段长度和你需要计算t秒之后小球的位置。 　　第二行包含n个整数a1, a2, …, an，用空格分隔，表示初始时刻n个小球的位置。  **输出格式**  　　输出一行包含n个整数，用空格分隔，第i个整数代表初始时刻位于ai的小球，在t秒之后的位置。  **样例输入**  3 10 5 4 6 8  **样例输出**  7 9 9  **样例说明**  　　初始时，三个小球的位置分别为4, 6, 8。 http://118.190.20.162/RequireFile.do?fid=b6beJN6e 　　一秒后，三个小球的位置分别为5, 7, 9。 http://118.190.20.162/RequireFile.do?fid=Ab8QmfeR 　　两秒后，第三个小球碰到墙壁，速度反向，三个小球位置分别为6, 8, 10。 http://118.190.20.162/RequireFile.do?fid=fgQLYbNn 　　三秒后，第二个小球与第三个小球在位置9发生碰撞，速度反向（注意碰撞位置不一定为偶数），三个小球位置分别为7, 9, 9。 http://118.190.20.162/RequireFile.do?fid=erfyNJDT 　　四秒后，第一个小球与第二个小球在位置8发生碰撞，速度反向，第三个小球碰到墙壁，速度反向，三个小球位置分别为8, 8, 10。 http://118.190.20.162/RequireFile.do?fid=m5EBf6q8 　　五秒后，三个小球的位置分别为7, 9, 9。 http://118.190.20.162/RequireFile.do?fid=JRHaHt4T  **样例输入**  10 22 30 14 12 16 6 10 2 8 20 18 4  **样例输出**  6 6 8 2 4 0 4 12 10 2  **数据规模和约定**  　　对于所有评测用例，1 ≤ n ≤ 100，1 ≤ t ≤ 100，2 ≤ L ≤ 1000，0 < ai < L。L为偶数。 　　保证所有小球的初始位置互不相同且均为偶数。 |

**三、URL映射**

|  |  |
| --- | --- |
| 试题编号： | 201803-3 |
| 试题名称： | URL映射 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　URL 映射是诸如 Django、Ruby on Rails 等网页框架 (web frameworks) 的一个重要组件。对于从浏览器发来的 HTTP 请求，URL 映射模块会解析请求中的 URL 地址，并将其分派给相应的处理代码。现在，请你来实现一个简单的 URL 映射功能。 　　本题中 URL 映射功能的配置由若干条 URL 映射规则组成。当一个请求到达时，URL 映射功能会将请求中的 URL 地址按照配置的先后顺序逐一与这些规则进行匹配。当遇到第一条完全匹配的规则时，匹配成功，得到匹配的规则以及匹配的参数。若不能匹配任何一条规则，则匹配失败。 　　本题输入的 URL 地址是以斜杠 / 作为分隔符的路径，保证以斜杠开头。其他合法字符还包括大小写英文字母、阿拉伯数字、减号 -、下划线 \_ 和小数点 .。例如，/person/123/ 是一个合法的 URL 地址，而 /person/123? 则不合法（存在不合法的字符问号 ?）。另外，英文字母区分大小写，因此 /case/ 和 /CAse/ 是不同的 URL 地址。 　　对于 URL 映射规则，同样是以斜杠开始。除了可以是正常的 URL 地址外，还可以包含参数，有以下 3 种： 　　字符串 <str>：用于匹配一段字符串，注意字符串里不能包含斜杠。例如，abcde0123。 　　整数 <int>：用于匹配一个不带符号的整数，全部由阿拉伯数字组成。例如，01234。 　　路径 <path>：用于匹配一段字符串，字符串可以包含斜杠。例如，abcd/0123/。 　　以上 3 种参数都必须匹配非空的字符串。简便起见，题目规定规则中 <str> 和 <int> 前面一定是斜杠，后面要么是斜杠，要么是规则的结束（也就是该参数是规则的最后一部分）。而 <path> 的前面一定是斜杠，后面一定是规则的结束。无论是 URL 地址还是规则，都不会出现连续的斜杠。  **输入格式**  　　输入第一行是两个正整数 *n* 和 *m*，分别表示 URL 映射的规则条数和待处理的 URL 地址个数，中间用一个空格字符分隔。 　　第 2 行至第 *n*+1 行按匹配的先后顺序描述 URL 映射规则的配置信息。第 *i*+1 行包含两个字符串 *pi* 和 *ri*，其中 *pi* 表示 URL 匹配的规则，*ri* 表示这条 URL 匹配的名字。两个字符串都非空，且不包含空格字符，两者中间用一个空格字符分隔。 　　第 *n*+2 行至第 *n*+*m*+1 行描述待处理的 URL 地址。第 *n*+1+*i* 行包含一个字符串 *qi*，表示待处理的 URL 地址，字符串中不包含空格字符。  **输出格式**  　　输入共 *m* 行，第 *i* 行表示 *qi* 的匹配结果。如果匹配成功，设匹配了规则 *pj* ，则输出对应的 *rj*。同时，如果规则中有参数，则在同一行内依次输出匹配后的参数。注意整数参数输出时要把前导零去掉。相邻两项之间用一个空格字符分隔。如果匹配失败，则输出 404。  **样例输入**  5 4 /articles/2003/ special\_case\_2003 /articles/<int>/ year\_archive /articles/<int>/<int>/ month\_archive /articles/<int>/<int>/<str>/ article\_detail /static/<path> static\_serve /articles/2004/ /articles/1985/09/aloha/ /articles/hello/ /static/js/jquery.js  **样例输出**  year\_archive 2004 article\_detail 1985 9 aloha 404 static\_serve js/jquery.js  **样例说明**  　　对于第 1 个地址 /articles/2004/，无法匹配第 1 条规则，可以匹配第 2 条规则，参数为 2004。 　　对于第 2 个地址 /articles/1985/09/aloha/，只能匹配第 4 条规则，参数依次为 1985、9（已经去掉前导零）和 aloha。 　　对于第 3 个地址 /articles/hello/，无法匹配任何一条规则。 　　对于第 4 个地址 /static/js/jquery.js，可以匹配最后一条规则，参数为 js/jquery.js。  **数据规模和约定**  　　1 ≤ *n* ≤ 100，1 ≤ *m* ≤ 100。 　　所有输入行的长度不超过 100 个字符（不包含换行符）。 　　保证输入的规则都是合法的。 |

**四、再卖菜**

|  |  |
| --- | --- |
| 试题编号： | 201803-4 |
| 试题名称： | 棋局评估 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　Alice和Bob正在玩井字棋游戏。 　　井字棋游戏的规则很简单：两人轮流往3\*3的棋盘中放棋子，Alice放的是“X”，Bob放的是“O”，Alice执先。当同一种棋子占据一行、一列或一条对角线的三个格子时，游戏结束，该种棋子的持有者获胜。当棋盘被填满的时候，游戏结束，双方平手。 　　Alice设计了一种对棋局评分的方法： 　　- 对于Alice已经获胜的局面，评估得分为(棋盘上的空格子数+1)； 　　- 对于Bob已经获胜的局面，评估得分为 -(棋盘上的空格子数+1)； 　　- 对于平局的局面，评估得分为0；  http://118.190.20.162/RequireFile.do?fid=yrNHga4H 　　例如上图中的局面，Alice已经获胜，同时棋盘上有2个空格，所以局面得分为2+1=3。 　　由于Alice并不喜欢计算，所以他请教擅长编程的你，如果两人都以最优策略行棋，那么当前局面的最终得分会是多少？  **输入格式**  　　输入的第一行包含一个正整数*T*，表示数据的组数。 　　每组数据输入有3行，每行有3个整数，用空格分隔，分别表示棋盘每个格子的状态。0表示格子为空，1表示格子中为“X”，2表示格子中为“O”。保证不会出现其他状态。 　　保证输入的局面合法。(即保证输入的局面可以通过行棋到达，且保证没有双方同时获胜的情况) 　　保证输入的局面轮到Alice行棋。  **输出格式**  　　对于每组数据，输出一行一个整数，表示当前局面的得分。  **样例输入**  3 1 2 1 2 1 2 0 0 0 2 1 1 0 2 1 0 0 2 0 0 0 0 0 0 0 0 0  **样例输出**  3 -4 0  **样例说明**  　　第一组数据： 　　Alice将棋子放在左下角(或右下角)后，可以到达问题描述中的局面，得分为3。 　　3为Alice行棋后能到达的局面中得分的最大值。 　　第二组数据： http://118.190.20.162/RequireFile.do?fid=fEF6hem5  　　Bob已经获胜(如图)，此局面得分为-(3+1)=-4。 　　第三组数据： 　　井字棋中若双方都采用最优策略，游戏平局，最终得分为0。  **数据规模和约定**  　　对于所有评测用例，1 ≤ *T* ≤ 5。 |

**五、二次求和**

|  |  |
| --- | --- |
| 试题编号： | 201803-5 |
| 试题名称： | 二次求和 |
| 时间限制： | 10.0s |
| 内存限制： | 512.0MB |
| 问题描述： | **问题描述**  　　给一棵 n 个节点的树，用 1 到 n 的整数表示。每个节点上有一个整数权值 ai。再给出两个整数 L,R。现在有 m 个操作，每个操作这样描述： 　　给定树上两个节点 u,v 和一个整数 d，表示将树上 u 到 v 唯一的简单路径上每个点的权值 ai 都加上 d。之后求树上所有节点个数大于等于 L 小于等于 R 的简单路径的节点权值和之和。注意这里有两次求和：对于一条节点个数大于等于 L 小于等于 R 的简单路径，求出它所有节点的权值之和；然后对所有这样的路径，对它们的权值和再进行求和。因为答案很大，只用输出对Q=1,000,000,007 取余的结果即可。  **输入格式**  　　从标准输入读入数据。 　　包含多组数据。数据的第一行包含一个正整数 T，表示数据组数。保证 T=10。 　　每组数据的第一行包含 4 个非负整数 n,m,L,R，分别表示节点个数、操作个数和询问相关的两个参数。保证 1≤L≤R≤n。 　　第二行包含 n 个整数，表示 ai。保证 0≤ai<Q。 　　第三行包含 n-1 个整数，描述树的形态，其中第 i 个数 fi 表示节点 i+1 与节点 fi 之间有一条边，保证 1≤fi≤i<n。能够证明，这样能够保证给出的是一棵树。 　　接下来 m 行，每行包含三个整数 u,v,d，描述一个操作即将树上 u 到 v 唯一的简单路径上每个点的权值 ai 都加上 d，之后询问满足条件的两次求和的结果。保证 1≤u,v≤n，0≤d<Q。 　　保证上述同一行所有的数之间，均用一个空格隔开。  **输出格式**  　　输出到标准输出。 　　输出 m 行每行一个整数，表示两次求和的结果对 Q=1000000007 求余的结果。 　　子任务 　　共有 10 个测试点，各测试点特点如下： 　　测试点1：n=10，m=10。 　　测试点2：n=50，m=50。 　　测试点3：n=300，m=300。 　　测试点4：n=2000，m=2000。 　　测试点5：n=2000，m=100000。 　　测试点6：n=100000，m=100000。保证 fi=i。 　　测试点7：n=100000，m=100000。保证 fi=⌊(i+1)/2⌋，其中 ⌊⋅⌋ 表示向下取整，即给定的树是完全二叉树。 　　测试点8：n=100000，m=100000。保证对于同一组数据，相同的 fi 至多出现 2 次，即给定的树是二叉树。 　　测试点9：n=100000，m=100000。 　　测试点10：n=100000，m=100000。 　　上述约束对同一测试点中的每组数据都有效。  **样例输入**  10 10 10 3 6 36 11 76 24 71 89 24 63 75 40 1 1 2 2 3 3 4 4 5 2 5 18 5 7 95 7 10 82 8 2 99 8 1 85 7 7 60 1 5 85 4 3 38 9 4 17 1 1 99 10 10 5 9 33 93 91 43 82 56 14 85 30 56 1 2 3 4 5 6 7 8 9 9 3 26 1 4 7 6 3 36 3 2 32 4 8 83 8 2 84 9 5 89 3 1 84 4 4 86 1 10 67 10 10 3 6 89 35 24 21 76 48 24 20 58 23 1 2 2 2 1 6 7 7 7 6 10 56 2 1 9 4 9 62 1 7 64 1 6 53 2 10 6 2 7 87 4 3 53 1 7 61 2 5 28 10 10 3 5 59 2 90 23 45 75 30 42 29 19 1 2 2 2 1 6 7 7 7 8 2 14 9 3 66 2 8 46 2 7 33 8 10 57 8 8 2 10 6 50 10 6 19 1 9 80 1 5 60 10 10 3 6 42 47 99 53 40 62 36 93 55 15 1 2 2 2 1 6 7 7 7 2 5 82 6 2 25 10 7 76 1 4 59 9 4 7 9 3 34 9 4 29 6 2 41 3 10 13 6 7 79 10 10 3 5 56 99 67 3 27 4 23 95 15 44 1 2 2 2 1 5 7 7 7 10 9 70 2 9 33 2 4 81 7 6 13 7 3 61 7 6 21 5 8 85 7 2 88 5 6 21 8 6 6 10 10 2 5 48 14 53 28 19 31 28 15 26 18 1 1 3 3 1 6 1 8 8 9 6 96 2 9 97 2 3 27 2 3 72 8 7 96 4 10 63 8 7 10 3 9 45 3 10 76 4 5 93 10 10 2 4 26 66 62 35 24 86 52 96 67 47 1 1 3 3 1 6 1 8 8 7 8 4 7 8 99 9 6 71 8 4 58 5 1 37 3 9 9 7 10 74 10 9 42 7 7 71 7 2 0 10 10 2 5 65 10 44 20 2 11 11 31 39 15 1 1 3 3 1 6 1 8 8 2 8 69 7 2 27 2 9 65 1 2 74 4 8 78 7 5 98 5 2 21 5 2 34 3 8 85 10 4 40 10 10 2 4 7 81 18 6 64 36 79 46 97 53 1 1 3 3 1 6 1 8 8 10 5 10 2 3 5 1 2 44 3 10 32 5 9 77 10 3 73 8 3 36 6 6 72 1 7 0 1 5 0  **样例输出**  7591 17186 26124 32163 39473 39953 46073 49835 50328 52803 10590 10912 13396 14164 21302 30542 37573 40009 41471 50181 10425 10902 18466 23586 26448 27132 36354 38580 43460 44412 5246 10526 13976 16286 17825 17835 19835 20595 25235 27635 10594 12594 15178 18777 19631 23779 27317 30597 32183 36370 8770 11443 14278 15461 20463 22374 26964 33476 34904 35492 13628 21388 23305 28417 36865 43417 44297 48572 55792 59698 6898 13432 18189 22365 24363 25011 30339 31599 31954 31954 10229 12200 17400 20878 28288 37794 39474 42194 49504 53664 5054 5339 7055 9359 15365 20621 22997 23933 23933 23933  **样例输入**  10 10 20 3 6 491392628 636332471 19224378 895707216 212813471 660436885 742024058 218835312 940926800 569228205 1 1 2 2 3 3 4 4 5 8 7 521081694 8 1 516314553 8 5 161674128 9 7 969050697 8 5 880349526 4 4 721331815 8 9 505714465 5 5 336316695 9 7 128440701 8 6 99354882 8 2 254449937 6 9 370034609 10 1 903883022 6 3 680143641 6 6 619492963 9 2 68417811 3 9 909965675 6 6 180290971 6 6 24683608 9 7 729188440 10 20 5 9 943002276 444023513 720772812 822038827 6915379 969113777 988091028 592878322 291476680 947319722 1 2 3 4 5 6 7 8 9 4 5 348252821 1 6 249606044 8 6 62191899 8 7 358148284 6 4 242642175 6 7 637739010 6 3 765212619 7 5 556454511 3 8 756315861 6 6 728202502 10 6 285199168 2 10 378200272 1 6 875633316 6 6 775752986 2 2 398595378 10 5 331595217 3 2 831372304 9 4 664441394 9 5 782352279 3 10 510936096 10 20 3 6 204252620 587141701 608106462 848982202 858964458 605138352 33959841 265841584 245041584 847950362 1 2 2 2 1 6 7 7 7 8 10 571834650 7 6 676194195 6 3 708659 5 5 212535538 3 1 24749194 10 3 529132357 8 2 686536556 1 7 931180973 1 8 747624721 7 10 857820531 1 8 234369066 8 4 876243996 5 8 9419476 1 3 520362020 8 10 668522479 4 2 269454181 9 10 323592999 7 6 861845979 2 6 610344496 10 10 506977188 10 20 3 5 693867342 70636645 633468226 523071166 458972263 827120047 447544344 160061783 674855982 73755912 1 2 2 2 1 6 7 7 7 1 10 144596785 3 1 591878180 7 8 221928636 8 4 731308878 5 7 424754647 8 10 121922750 4 2 147606026 3 6 501836296 2 2 335053627 4 3 920195904 6 10 896875521 10 2 326960264 3 5 448136606 6 3 686320413 10 5 105646568 5 4 532789020 6 10 91140656 3 4 697465060 6 6 313064764 3 5 498351699 10 20 3 6 176956446 213334429 102569431 54270163 148010144 557816692 532831128 563116304 104041892 122092745 1 2 2 2 1 6 7 7 7 9 7 645536933 5 6 331374192 4 6 325430362 9 10 944136912 5 9 377118754 6 3 65718140 4 1 798214633 6 8 391320936 6 2 97366442 6 7 862519448 3 2 871809907 8 9 627269894 9 6 209346433 9 3 711333486 8 7 990297028 7 8 223638052 1 8 569218523 6 2 204569660 4 7 566890294 9 7 308286492 10 20 3 5 516866543 114802703 232002958 90418885 779481077 838174697 697434585 6723130 803141124 935649787 1 2 2 2 1 6 7 7 7 2 5 630775943 3 4 870019290 6 1 516445175 6 2 104778756 10 8 804804515 8 1 715359294 7 4 222275306 8 2 173241198 3 2 53498074 3 2 247915656 1 8 825785408 10 4 514272010 6 5 1205426 6 6 881910972 8 6 253040479 5 10 282628716 3 2 535404021 5 8 976667831 3 8 532657657 5 9 912002333 10 20 2 5 344753993 454884255 681358461 359962443 379128568 742537263 837324893 822811562 710479503 727754509 1 1 3 3 1 6 1 8 8 6 7 780482512 9 7 431968304 7 3 530362547 7 6 209957540 10 4 108463764 2 2 509079737 3 6 535472535 9 8 902194399 1 7 558093699 9 10 886419834 10 3 723007263 9 2 930488874 5 5 312267119 5 2 781360758 4 2 79490779 2 2 136304873 2 4 154642800 7 7 192682645 10 9 780239411 5 7 989608025 10 20 2 4 988984729 204292220 405708381 936488116 353640506 157175515 362805513 804969086 537282381 260345075 1 1 3 3 1 6 1 8 8 6 1 748160253 7 1 430491219 5 3 378043377 6 4 929835633 3 9 707534391 7 2 128270698 4 10 538763858 10 4 251763733 9 2 194241728 9 1 788115489 1 3 676299601 7 10 938741480 6 10 59357319 2 7 817843160 6 3 40182504 5 10 847865792 9 6 96314986 9 4 49220393 4 2 239362285 3 3 446518941 10 20 2 5 112829999 189088440 137145976 251843687 640426363 731293400 7360709 218921466 944422335 548980830 1 1 3 3 1 6 1 8 8 2 7 348698740 3 1 618099270 10 5 999382450 9 1 224374017 5 6 687839717 4 9 245105135 3 1 325959540 2 5 273980999 9 8 426858352 9 3 106778086 6 7 932703346 6 9 184228384 9 7 249483657 10 7 599742376 9 10 628359618 3 4 577643945 3 9 726254383 10 1 345942803 4 1 744803967 6 10 139462432 10 20 2 4 355725276 663365076 571961759 37770117 16632744 156046075 518924620 889404943 71869448 99325628 1 1 3 3 1 6 1 8 8 10 3 345164780 6 7 857748538 10 1 748140085 9 10 104973573 2 6 521974741 10 2 375746598 2 9 970508670 10 1 580779856 1 3 138320042 6 5 678098079 7 9 909097431 8 5 25624210 1 7 139991385 7 10 963577297 10 3 516128436 4 3 725558635 10 8 752504151 6 9 592891952 10 5 630022989 6 6 578601365  **样例输出**  29224221 432275471 719509115 160338486 66901993 214870003 926305082 971055472 741735982 167547328 688993380 242973114 553614370 277779819 233723488 407209931 773536477 215864231 413333095 270003107 51590172 18497721 128092650 230689377 576008911 534613110 334283450 939281345 570866913 406714353 944660147 219693811 772891844 512198473 498152225 352150257 305085413 91458789 897288403 654939030 418344215 256636298 318998290 19282580 528983407 83130506 348297344 842774666 633749652 799647503 424125164 325891927 475067992 217150988 295094910 456537001 47442861 725279433 552838770 608656246 194389606 869516645 751946602 256656429 113254730 405168959 652501510 759006475 454918092 300207325 175227913 697247545 796935823 603519497 55244874 440548316 86174528 917731022 552896732 8392507 940239879 101168565 739040225 392790249 401277915 184474193 875566470 746143398 535458702 248989124 890525759 235861118 5993440 788678130 458776844 62470556 153700230 519272918 144765979 626506637 312931305 803451974 395478141 948752174 678473925 169312683 839960521 833050280 10007894 464152291 359705619 501466132 571380840 445778224 567397314 177694433 956582818 90008745 702621011 662807140 342217817 243143011 915046825 373942823 654174202 235891800 538221771 310636728 28633212 258265981 943955490 383064885 193468935 702329141 61591412 288335262 659759178 393902969 163958000 155935753 855102156 518680521 591721506 890708483 833184278 144614008 168194638 805765679 42994452 601230564 63611181 652997272 629937617 246997408 698130138 831661452 284765465 123956091 203779941 241120823 959255155 281409622 217183694 147738789 677633465 168567323 378058663 296538429 382863947 526782047 777068875 989166555 189081109 364091175 755194949 817445001 811610903 373549741 254631027 527324959 775317946 214791518 614355828 763562997 906249340 578284846 720330629 82442629 721804603 154375574 609390151 454333257 173919688 551484589 712731722 126138836 186238334 909998845 51791637 573609333 |