**A. 约瑟夫问题**

**单点时限:**2.0 sec

**内存限制:**512 MB

“约瑟夫问题，怎么又是约瑟夫问题。” 我嘟囔着，心里却开始盘算阴谋。

小学的开始，学数组的时候用 Pascal 写约瑟夫问题。

初中的时候，学模拟的时候用 Pascal 写约瑟夫问题。

高中的时候，学指针的时候用 C++ 写约瑟夫问题。

本科的时候，学算法的时候用 Python 写约瑟夫问题。

而现在，考场上，约瑟夫问题卷土重来，却好似没了以前的面目。

你抱怨着从来没有听说过这个问题，所以我只能再给你讲一遍什么是约瑟夫问题：

N 个人围成一圈，每个人按照顺时针的顺序，标号为 1 到 N 。现在从标号为 1 的人开始按照顺时针的顺序报数，数到 M 的人出圈；再由下一个人从 1 开始重新报数，数到 M 的人出圈（需要注意的是，出圈的人将不在参与之后的报数）......

你说你还是没有很清楚，所以我决定给你举一个例子：

5 个人围成一个圈，每次报数为 2 的人将出圈。

| **报数序号** | **标号** | **所报的数** | **是否出圈** |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 否 |
| 2 | 2 | 2 | 是 |
| 3 | 3 | 1 | 否 |
| 4 | 4 | 2 | 是 |
| 5 | 5 | 1 | 否 |
| 6 | 1 | 2 | 是 |
| 7 | 3 | 1 | 否 |
| 8 | 5 | 2 | 是 |
| 9 | 3 | 1 | 否 |
| 10 | 3 | 2 | 是 |

所以出圈的顺序为：2、4、1、5、3。

讲到这里，我的阴谋也基本成型了。现在，我会告诉你围成圈的一共有 N 个人，每次数到 M 的人将出圈，你需要告诉我的是，第 K 个出圈的人的标号。

**输入格式**

数据第一行包含一个整数 T ，表示测试数据组数。

对于每一组测试数据，包含一行三个整数 N,M,K ，之间用空格隔开，意义参见题面。

**输出格式**

对于每组测试数据输出一行包含一个整数，表示答案。

**样例**

**input**

5

5 2 1

5 2 2

5 2 3

5 2 4

5 2 5

**output**

2

4

1

5

3

**input**

5

10 3 5

10 5 2

10 1 8

10 8 10

5 6 1

**output**

7

10

8

1

1

**提示**

**对于数据范围的约定：**

* 对于50%的数据，保证 1≤K≤N≤100,1≤M≤100,1≤T≤50 ；
* 对于80%的数据，保证 1≤K≤N≤105,1≤M≤105,1≤T≤50 ；
* 对于100%的数据，保证 1≤K≤N≤1018,1≤M≤1018,1≤T≤50 且 min{M,K}≤106 。

**B. 合法变量**

**单点时限:**1.0 sec

**内存限制:**256 MB

马上就要成为研究生了。我决定考考你对基础知识的掌握程度。

我会给你一个不包含空格或者其他空白字符的字符串，请判断是否是 C 语言合法的变量名称。

我保证给你的字符串一定不是 C 语言的保留字。

如果一个字符串是 C 语言合法的变量名称，必须满足以下要求：

* 非保留字。
* 只包含字母、数字及下划线（“\_”）。
* 不以数字开头。

**输入格式**

输入包含一行一个不包含空格或者其他空白字符的字符串，保证字符串的长度不超过 30 。

**输出格式**

一行，如果它是 C 语言的合法标识符，则输出 “yes” ，否则输出 “no” 。

**样例**

**input**

iloveyou

**output**

yes

**input**

555\_failed

**output**

no

窗体顶端

**C. 晚饭吃什么**

**单点时限:**1.0 sec

**内存限制:**256 MB

又到了吃晚饭的时间， dx 又来到了熟悉的二楼，面对着众多美食她的选择困难症又犯了。

这时 xxh 提出一个建议，食堂一共有 m 个窗口，将它们按照 1,2,3,⋯,m 来排序。由 xxh 来选择一个数字 n ， dx 按照 1,2,3,⋯,m,1,2,⋯ 的顺序，直到第 n 个窗口就是 dx 今晚点餐的窗口。

例：食堂一共有 7 个窗口， xxh 给出数字 9，则 dx 将在 2 号窗口点餐。

dx 一口就答应了，但没想到 xxh 给的数字实在是太大了。 dx 根本数不过来，但又不甘心就这样放弃，请想想办法帮助 dx 找到她今晚点餐的窗口编号。

**输入格式**

输入数据第一行包含一个整数 T ( 1≤T≤1000 )，表示数据组数。

接下来 T 行，每行包含两个整数 n,m ( 2≤n≤101000,2≤m≤108 )，表示所选的数字和窗口数量。

**输出格式**

输出每行包含一个整数，表示 dx 今晚点餐的窗口编号。

**样例**

**input**

2

148 90

183 50

**output**

58

33

**input**

5

7660882 2070

38198 9970

4287 74680

6108 3957955

76712 15

**output**

1882

8288

4287

6108

2

**提示**

对于10%的数据保证 2≤n≤109 ；

对于30%的数据保证 2≤n≤1018 ；

对于100%的数据保证 2≤n≤101000 。

**D. 津津的数字压缩法**

**单点时限:**2.0 sec

**内存限制:**512 MB

最近津津需要处理一大堆长长的数字，可是她却买不起大容量的 SSD 来存放这些数据。

在一顿深入思考、冷静分析后，津津想出来了这样一个压缩数字的方法：

* 譬如说，数字 1111122 可以被描述为“ 5 个 1 和 2 个 2 ”，所以可以被压缩为 5122 ；
* 又譬如， 122344111 可被描述为” 1 个 1 、 2 个 2 、 1 个 3 、 2 个 4 和 3 个 1 “，所以可以被压缩为 1122132431 。

类似的道理， 1111111111 可以压缩为 101 ， 00000000000 可压缩为 110 ， 100200300 可压缩为 112012201320 。

当然在很多情况下这样子压缩反而让数字边长了…但是津津才不管这些呢！

所以我们的任务是帮助津津按照她的压缩方法压缩她给出的数据，毕竟她开心就好。

**输入格式**

输入仅一行，包含一个很长很长的非负整数 s （可能会出现前导 0 ）。

数据保证 s 的位数在 1 到 1000 之间。

**输出格式**

一行，对应的压缩结果。

**样例**

**input**

122344111

**output**

1122132431

**input**

00000000000

**output**

110

**E. 财务危机**

**单点时限:**1.0 sec

**内存限制:**256 MB

小啵同学毕业之后决定创业。他招聘了 N 名员工，并且为每名员工都配备了一个办公室。

好景不长，仅仅过了一个月，公司就陷入了财务危机。无奈之下，小啵决定只留下 M 个办公室。这也就意味着，所有的员工需要重新安排办公室。

小啵想知道有多少种不同的安排方法。

请注意，小啵不允许有办公室空出来（否则他就可以租更少的办公室）；办公室没有编号；两个方案被认为是相同的，当且仅当在两个方案中，每个人的办公室同事都是相同的。

具体请参考样例解释。

**输入格式**

输入数据仅有一行，包含两个整数N和M，表示员工数量和留下办公室的个数。

**输出格式**

输出一个整数P，代表所有的方案数。

答案可能很大，你只需要对答案模1000000007(109+7)输出即可。

**样例**

**input**

3 2

**output**

3

**input**

4 3

**output**

6

**提示**

在第一组样例中有3种方案：（1，23）（2，13）（3，12）  
在第二组样例中有6种方案：（1，2，34）（1，3，24）（1，4，23）（2，3，14）（2，4，13）（3，4，12）

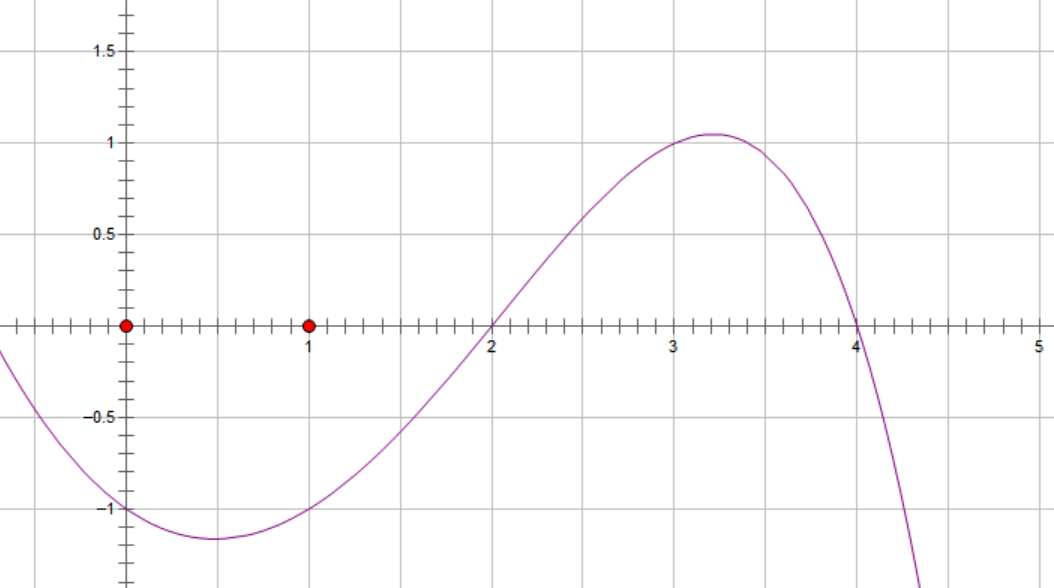
小啵同学为了不让你为难，偷偷给了你一些提示：  
对于 50% 的数据，保证 1≤N≤10   
对于 80% 的数据，保证 1≤N≤103   
对于 100% 的数据，保证 1≤N≤106   
对于所有数据均保证：0≤M≤103且M≤N

**F. 超越方程**

**单点时限:**3.5 sec

**内存限制:**256 MB

求超越函数 f(x)=k∗xp−mx ，其中 k,m,p∈Z+,p≥2,m≥2 ,在闭区间中 [a,b] 的最大值。



当然，由于超越函数不具有良好的函数特性，比如：超越函数不一定是单调函数。

于是，为了简化问题，数据中保证 f(x) 在闭区间 [a,b] 中是一个单峰函数，也就是说函数 f(x) 在区间 [a,b] 上只有唯一的最大值点 C ，而在最大值点 C 的左侧，函数单调增加；在点 C 的右侧，函数单调减少。

是不是感觉很简单呢？

**输入格式**

第一行一个整数 t ，表示测试组数。

接下来 t 行，每行五个整数 k,m,p,a,b 。

**输出格式**

共 t 行，每行一个小数 x0 满足 a<x0<b ，且满足 f(x0) 是 f(x) 在闭区间 [a,b] 上的最大值。

你的答案和标准答案的绝对误差或相对误差在 10−6 之内会被认为是正确的。

**样例**

**input**

1

1 2 2 2 4

**output**

3.21243252

**提示**

**温馨提示**

1、long double 最大范围约在104932，long double 输出请用prinf(“%.8Lf\n”,ans);或者强制类型转换成double都输出。（for c & c++）。

2、double 最大范围约在10308，double 输出请用prinf(“%.8lf\n”,ans);（for c & c++）。

3、java 请使用 java.math.BigDecimal ，

4、读入数据量大，请使用 scanf（for c & c++） 。

5、请一定要认真阅读数据约定！！！！！

**数据约定**

30% 数据满足 t=1,1≤k≤10,2≤m≤10,2≤p≤10,0≤a<b≤10 ；

50% 数据满足 1≤t≤10,1≤k≤100,2≤m≤10,2≤p≤100,0≤a<b≤103 ；

70% 数据满足 1≤t≤103,1≤k≤100,2≤m≤10,2≤p≤100,0≤a<b≤103 ；

100% 数据满足 1≤t≤104,1≤k≤106,2≤m≤103,2≤p≤106,0≤a<b≤107 。

**G. 找数**

**单点时限:**1.0 sec

**内存限制:**256 MB

输入一个整数 n(2≤n≤10) ，你需要找到一些 n 位数（允许有前置 0 ，见样例），这些 n 位数均由 0 ~ n−1 这些数字组成。

并且每个数字恰好只出现一次。此外，这个 n 位数中前 n2 位数组成的数恰好是后 n2 位数组成的数的整数倍。按从小到大的顺序输出所有满足条件的 n 位数。

**输入格式**

一个整数 n ( 2≤n≤10 且 n 为偶数）。

**输出格式**

每行输出一个 n 位数。

表示满足条件的所有 n 位数，按升序排序。

**样例**

**input**

2

**output**

01

窗体顶端