

BCSP小高组编程能力题目样卷

题目名称	7 的倍数	计算器	放学路径	数字键盘
题目类型	传统型	传统型	传统型	传统型
目录	seven	calc	school	keyboard
可执行文件名	seven	calc	school	keyboard
输入文件名	seven.in	calc.in	school.in	keyboard.in
输出文件名	seven.out	calc.out	school.out	keyboard.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒	1.0 秒
内存限制	256 MiB	256 MiB	512 MiB	512 MiB
测试点数目	10	20	10	49
测试点是否等分	是	是	是	否

提交源程序文件名

对于 C++ 语言	seven.cpp	calc.cpp	school.cpp	keyboard.cpp
-----------	-----------	----------	------------	--------------

编译选项

对于 C++ 语言	-O2 -std=c++14
-----------	----------------

T1 7 的倍数（seven）

题目描述

一周只有 7 天，并且每周都有周六或周日可以休息。

小 Z 因此对 7 这个数字产生了兴趣。现在给定两个正整数 n 和 m ，小 Z 可以从区间 $[1, n]$ 和区间 $[1, m]$ 中分别选出两个整数 x, y ，即选出来的 x, y 需要满足 $1 \leq x \leq n, 1 \leq y \leq m$ 。

问，现在有多少对不同的 (x, y) 的组合使得 $x + y$ 是 7 的倍数。

这里，不同的组合指的是，对于任意一对 (x, y) 只要对应的 x 或 y 不同就算不同。

输入格式

从 `seven.in` 文件读入数据。

一行，输入两个正整数 n, m 。

输出格式

输出到 `seven.out` 文件。

输出一行一个整数表示答案。

样例

输入数据1

```
1 | 6 7
```

输出数据1

```
1 | 6
```

说明/提示

样例解释

选出的数对可以是 $(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)$ 。

数据范围

对于 30% 的数据， $1 \leq n, m \leq 10^2$ 。

对于所有的数据， $1 \leq n, m \leq 10^6$ 。

T2 计算器 (calc)

题目描述

小 Z 想自己做一个计算器，使得输入一个包含加法、减法、乘法三种运算的字符串算式，形如 `3+5*4*2+1=`，这个计算器就可以输出算式的计算结果。

当然众所周知，如果有大量乘法连在一起，那么很快这个数就会变得很大，故而我们只需要输出答案对 100000 的余数即可。

- 注意，本题计算取模时，负数的取模结果应该要为正数，即做减法时，取模应该是 $(a - b + p) \bmod p$ ，其中 p 是模数。
- 例如 `(1-3) % 3` 要写成 `(1-3+3) % 3` 他的结果应该是 1。

不过为了让题目尽可能地简单，对于输入的字符串，有如下保证：

- 字符串一定是一个个位数（0 ~ 9）开始，且个位数和运算符交替出现，最终以等号结尾。即数字不会连续出现。

- 运算符要么仅包含 `+` 和 `-`，要么仅包含 `+` 和 `*`。
- 综上所述，字符串中仅包含 `0 ~ 9`、`+`、`-`、`*` 共 13 种字符，且 `-` 和 `*` 不会同时出现。
- 字符串的长度 $length \in [2, 1000]$ 。

输入格式

从 `calc.in` 文件读入数据。

输入包含一行，为一个符合"保证"的字符串。

输出格式

输出到 `calc.out` 文件。

输出包含一行一个整数，代表输入算式的计算结果对 100000 取余的结果。

样例

输入数据1

```
1 | 1+2*3=
```

输出数据1

```
1 | 7
```

输入数据2

```
1 | 2+1-3+4-1+0-3=
```

输出数据2

```
1 | 0
```

说明/提示

对于 5% 的测试数据，满足输入中不包含字符 `+`、`-`、`*`；

对于 20% 的测试数据，满足输入中不包含字符 `-`、`*`；

对于 50% 的测试数据，满足输入中不包含字符 `*`；

对于所有测试数据，字符串中仅包含 `0~9`、`+`、`-`、`*` 这 13 种字符，保证 `-` 和 `*` 不会同时出现，字符串的长度 $length \in [2, 1000]$ 。

T3 放学路径 (school)

题目描述

小 Z 每天放学都步行回家，可以把从学校到家的路径视为 n 个方格排成一排，起点是第 1 个方格，终点是第 n 个方格。如下图所示。



小 Z 可以走四种步长的步伐，分别是向右走 1、2、3、4 格，小 Z 有轻微强迫症，他每一种步长的步伐都有规定的次数，且最终正好从 1 号方格走到 n 号方格。

现在小 Z 想走 m 步，从 1 号方格走到 n 号方格。路径上每一个格子都有一个快乐值，小 Z 会获得路径上所有的快乐值，他想知道获得的快乐值之和最大是多少。

输入格式

从 `school.in` 文件读入数据。

第 1 行两个正整数 n 和 m ，分别表示棋盘格子数和总步数。

第 2 行 n 个非负整数， $a_1, a_2, a_3, \dots, a_n$ ，其中 a_i 表示棋盘第 i 个格子的快乐值。

第 3 行 m 个整数， $b_1, b_2, b_3, \dots, b_m$ ，表示 m 个步伐的步长， b_i 为 1 至 4 之间的整数。(一定要用到 m 个步伐，顺序可以随意安排)

输入数据保证到达终点时刚好走了 m 步，即 $N - 1 = \sum_{i=1}^m b_i$ 。

输出格式

输出到 `school.out` 文件。

输出只有 1 行，1 个整数，表示小 Z 能获得的最大快乐值。

样例

输入数据1

1	9	5																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

输出数据1

1	73
---	----

输入数据2

7	8	9
4	5	6
1	2	3
0		

字键盘有一个光标。一开始，光标指向键盘上的 0 键。

小 Z 可以选择在一次操作中完成以下内容之一：

- 改变光标的指向。将光标的指向改为当前指向的键的相邻键，不能让光标指向没有键的位置。
- 输入光标所指的按键上的数字。

小 Z 想用数字键盘输入一个正整数 x ，使其除以 m 的余数为 r ，即需要满足 $x \bmod m = r$ 。

请你帮小 Z 找出输入这个正整数 x 所需的最少操作次数。

输入格式

从 `keyboard.in` 文件读入数据。

在一行中输入正整数 m, r ，含义如题所示。

输出格式

输出到 `keyboard.out` 文件。

输出一行一个整数表示满足要求的最少操作次数。

样例

输入数据1

```
1 | 100000 13
```

输出数据1

1 | 5

输入数据2

1 | 4 3

输出数据2

1 | 3

说明/提示

样例 1 解释

在这个例子中，可以通过执行以下五个操作来输入 13。

- 将光标向上移动，光标指向 1 键。
- 输入 1 。
- 将光标移到右边，光标指向 2 键。
- 将光标向右移动，光标指向 3 键。
- 输入 3 。到目前为止输入的数字变成 13。

所以输出 5。

样例 2 解释

在这个例子中，通过执行三次操作可以输入 11 。注意，输入 3 不是最佳选择，因为需要四次或更多的操作。

数据范围

子任务 1 有 30 分，满足 $m = 100000, 1 \leq r < m$

子任务 2 有 70 分，满足 $2 \leq m \leq 100000, 1 \leq r < m$