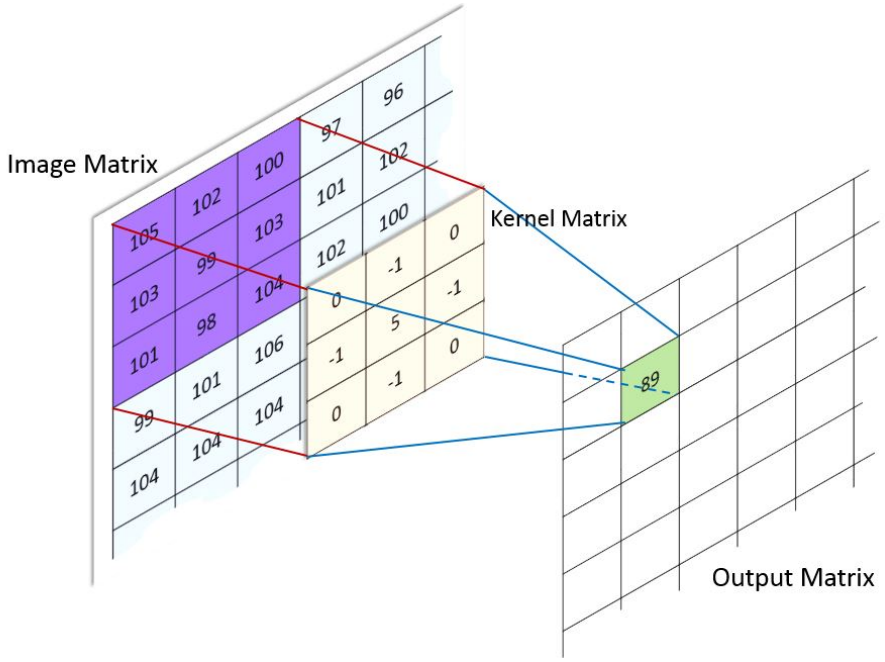
Problem A. 图像卷积

输入文件:	`standard input`
输出文件:	`standard output`
时间限制:	2 seconds per case
内存限制:	512 megabytes

如今机器学习热火朝天, 就算你的领域和机器学习八竿子打不着, 你也免不了会对机器学习中的一些概念和方法有所耳闻. 图像卷积技术可能就是非常有代表性的一个, 它被广泛地用于特征抽取, 是卷积神经网络 (CNN) 的核心组成部分.

如果你很遗憾地, 实在没有听说过图像卷积, 那也没有关系. 其实这个图像卷积本身, 跟机器学习也没啥关系. 它可以用下面的一张图来说明[1].



将 Kernel Matrix (核矩阵) 放在图像的左上角, 然后和原图像中的元素一一对应相乘并相加, 就得到了结果矩阵中的一个元素.

例如图中所示的

$$\mathbf{K} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

对应项相乘并相加得到 $0{\cdot}105+(-1){\cdot}102+0{\cdot}100+(-1){\cdot}103+5{\cdot}99+(-1){\cdot}103+0{\cdot}101+(-1){\cdot}98+0{\cdot}104 = 89$.

将 $\mathbf{K}$ 右移一个位置计算第二个元素: $0 \cdot 102 + (-1) \cdot 100 + 0 \cdot 97 + (-1) \cdot 99 + 5 \cdot 103 + (-1) \cdot 101 + 0 \cdot 98 + (-1) \cdot 104 + 0 \cdot 102 = 111$. 见图 1.

或许你会存在困惑说, 边缘的部分不就少了吗? 最后的结果矩阵会不会比原矩阵小? 有两种解决方案, 一种是在边缘填 0, 不够的部分当作 0 来计算; 还有一种是: 变小就变小好了. 此题中, 我们采用**第二种方案**.

[1] `http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html`

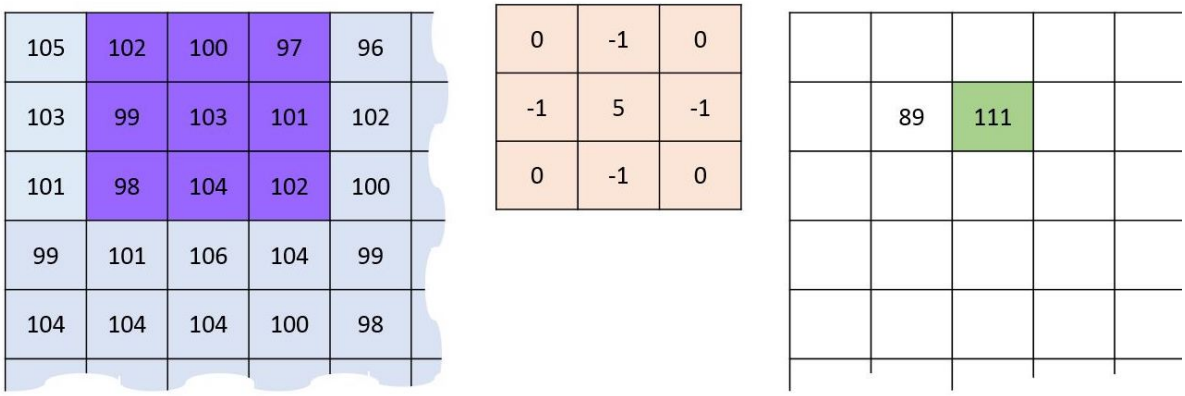


图 1: 卷积示例

输入

第一行两个整数 n, m.

接下来 n 行, 每行 m 个整数表示一个 $n \times m$ 的矩阵 $\mathbf{M}$, 这是原图像矩阵.

接下来一行两个整数 h, w.

接下来 h 行, 每行 w 个整数表示一个 $h \times w$ 的矩阵 $\mathbf{K}$, 这是核矩阵.

输出

输出卷积结果, 应是一个 $(n-h+1) \times (m-w+1)$ 的矩阵. 每行的元素之间用空格空开.

约束

- h, w, n, m 是整数; $1 \leqslant h \leqslant n \leqslant 100$; $1 \leqslant w \leqslant m \leqslant 100$;
- $\mathbf{M}_{ij}, \mathbf{K}_{ij}$ 是整数; $0 \leqslant \mathbf{M}_{ij} \leqslant 255$; $-128 \leqslant \mathbf{K}_{ij} \leqslant 127$.

子任务 1 (40 分)

$n = h$; $m = w$.

子任务 2 (60 分)

必须通过子任务 1. 否则不测试.

没有其他限制.

样例

standard input	standard output
5 5 105 102 100 97 96 103 99 103 101 102 101 98 104 102 100 99 101 106 104 99 104 104 104 100 98 3 3 0 -1 0 -1 5 -1 0 -1 0	89 111 101 85 111 101 98 117 113
2 3 1 1 2 3 5 8 2 3 13 21 34 55 89 -122	-264
3 2 0 1 2 3 4 5 1 2 -1 127	127 379 631

提示

本题在比对答案时忽略行末空格.

Problem B. 白网吧

输入文件:	`standard input`
输出文件:	`standard output`
时间限制:	3 seconds per case
内存限制:	512 megabytes

K 公司的老板 J 先生最近在华东师范大学开了一家网吧, 我们暂时把它叫作白网吧. 这网吧的生意, 说真的, 不咋地. 但这和 J 老板扩建白网吧的念头并不相干.

为了扩建网吧, J 老板首先想要了解网吧目前的经营状况. 现给出一段时间区间 (第 1 秒至第 m 秒, 共 m 秒) 内的 n 条上网记录 $(s_1, t_1); (s_2, t_2); \ldots; (s_n, t_n)$. s_i, t_i 分别表示第 i 条记录的进网吧时间和离开时间. **我们认为第 s_i 秒和第 t_i 秒这个人也在网吧内**.

针对这些数据, 回答 J 老板的问题.

输入

第一行两个整数 n, m.

接下来 n 行每行两个整数 s_i, t_i.

最后一行是一个整数 q. 表示问题类型.

输出

- 若 $q = 1$, 在一行内输出一个数, 表示这个网吧的最大同时在线人数.
- 若 $q = 2$, 输出平均在线人数. 平均在线人数用**最简分数**表示. (`p/q`: `p` 表示分子, `q` 表示分母. 分母为 `1` **不要省略**.)
- 若 $q = 3$, 先在一行内输出 $q = 1$ 的询问结果, 再在一行内输出 $q = 2$ 的询问结果.

约束

- $1 \leqslant n \leqslant 250\ 000$; $1 \leqslant m \leqslant 10^9$;
- $1 \leqslant s_i \leqslant t_i \leqslant m$;
- 同一对 (s_i, t_i) 可能出现多次, 看作是不同的上网记录;
- $q \in \{1, 2, 3\}$.

子任务 1 (28 分)

$1 \leqslant n \leqslant 10^3$; $q = 1$.

子任务 2 (24 分)

$1 \leqslant m \leqslant 100$; $q = 3$.

子任务 3 (32 分)

$q = 1$.

子任务 **4** (16 分)

$q = 2$.

样例

standard input	standard output
2 10 1 5 5 10 3	2 11/10
4 10 1 3 4 4 8 8 5 7 2	4/5
5 10 5 6 4 7 3 8 2 9 1 10 1	5

Problem C. 皇后问题

输入文件:	`standard input`
输出文件:	`standard output`
时间限制:	1 second per case
内存限制:	512 megabytes

有一个 $n \times m$ 的棋盘. 你现在的位置是 (x, y).

你是一个皇后. 在国际象棋规则中, 皇后是最强的棋子, 它可以向上、下、左、右、左上、右上、左下、右下八个方向移动任意多的空格 (只要还在棋盘内).

请设计一条路线, 用恰好 $nm - 1$ 步遍历棋盘上的所有格子. 当前的位置可以认为已经遍历过了.

输入

输入一行四个整数, 用空格隔开: n, m, x, y.

输出

输出答案序列. 共 $nm - 1$ 行, 每行是移动过后产生的新坐标, 同一行之间的两个整数用空格隔开:

$$
\begin{aligned}
&x_1\ y_1\\
&x_2\ y_2\\
&\vdots\\
&x_{nm-1}\ y_{nm-1}
\end{aligned}
$$

(x_i, y_i) 必须是合法的棋盘坐标, 即 $1 \leqslant x_i \leqslant n$, $1 \leqslant y_i \leqslant m$.

约束

- n, m, x, y 是整数;
- $1 \leqslant x \leqslant n \leqslant 100$; $1 \leqslant y \leqslant m \leqslant 100$;
- $mn > 1$.

子任务 1 (30 分)

$x = y = 1$.

子任务 2 (28 分)

必须先通过子任务 1.

$x = 1$.

子任务 3 (7 分)

必须先通过子任务 1.

$y = 1$.

子任务 4 (35 分)

必须先通过子任务 2, 3.

没有其他限制.

样例

standard input	standard output
3 3 1 1	3 3 1 3 2 3 2 1 3 1 2 2 1 2 3 2

解释

样例中的输出按照下面的路线. 0 表示起始位置, 1 到 8 依次表示经过的位置. 你可以自行验证该路线符合皇后的移动规则.

0	7	2
4	6	3
5	8	1

Problem D. 素数之基

输入文件:	`standard input`
输出文件:	`standard output`
时间限制:	1 second per case
内存限制:	512 megabytes

设有限**正整数集合** S, A (**不是多重集**, 每个数在集合中至多出现一次). 如果 S 中的每一个元素都是 A 的某一个子集之和, 我们称 A 是 S 的**基**.

形式化地说,

$$\forall x \in S, \exists A' \subseteq A, \text{such that} \sum_{a \in A'} a = x.$$

如果不存在 A' 也是 S 的基且 $|A'| < |A|$, 我们称 A 是 S 的**最小基**. 注意最小基不一定唯一.

$P(n)$ 是不超过 n 的所有素数的集合, 即 $P(n) = \{x | 1 \leqslant x \leqslant n, x\text{是素数}\}$. 求 $P(n)$ 的最小基.

如果有多解, 输出任意一解.

输入

一个正整数 n.

输出

输出第一行表示 $P(n)$ 的最小基包含多少个元素, 假设有 k 个.

第二行输出 k 个整数用空格隔开 $a_1, a_2, \ldots, a_k$. ($1 \leqslant a_i \leqslant 10^9$, a_i 各不相同).

评分

本题子任务有依赖关系, 在测试一个子任务首先必须通过在它之前的所有子任务 (除第 1 个子任务以外).

子任务 1 (10 分)

$n = 100$.

子任务 2 (20 分)

后面的子任务与第 1 个子任务独立.

$2 \leqslant n \leqslant 30$.

子任务 3 (16 分)

$31 \leqslant n \leqslant 42$.

子任务 4 (11 分)

$43 \leqslant n \leqslant 60$.

子任务 **5** (12 分)

$61 \leqslant n \leqslant 85$.

子任务 **6** (10 分)

$86 \leqslant n \leqslant 120$.

子任务 **7** (7 分)

$121 \leqslant n \leqslant 200$.

子任务 **8** (14 分)

$201 \leqslant n \leqslant 10^6$.

样例

standard input	standard output
2	1 2
5	2 2 3
10	3 2 1 4

解释

样例 3 解释: 10 以内的素数有 2, 3, 5, 7. 其中 $2 = 2$, $3 = 1 + 2$, $5 = 1 + 4$, $7 = 1 + 2 + 4$.

提示

如果你的程序花了几个小时跑出了结果的话, 你可以提交一个直接输出答案的程序. 但由于 OJ 限制, 你提交的程序长度不能超过 64KB.

Problem E. 痛苦的老爷机

输入文件:	`standard input`
输出文件:	`standard output`
时间限制:	4 seconds per case
内存限制:	512 megabytes

你的朋友唐纳德在你的生日聚会上送了你一台老爷机. 你严重怀疑该老爷机是外太空飞来的, 因为它是由一套叫作 Z86+ 的指令集驱动的, 并且其汇编语言 (如果真的叫汇编语言的话) 看起来很像是三地址代码[2].

你现在要写一段 Z86+ 能识别的程序实现**一个 32 位有符号整数数组的排序**. 由于这是一台老爷机, 它的运算速度、寄存器数量、内存空间大小都是非常受限的... 好吧, 你或许会觉得这样的设定很无聊, 但很不幸, 这就是问题的核心.

详细说明

首先说明可以使用的资源:

1. Z86+ 共有 A 个寄存器. 寄存器从 1 开始编号. 例如 $A = 4$ 时, 寄存器分别为 `r1`, `r2`, `r3`, `r4`.
2. 在本题中, 我们会在 Z86+ 中开辟两块内存空间. 第一块是 1 个字[3], 用于存储 n; 第二块是 M ($M \geqslant n$) 个字, 其中 $M[0], \ldots, M[n-1]$ 用于存储 a 数组. 你的程序不能也不该使用未开辟的内存区域. 错误使用会导致你的程序异常终止.
3. 该老爷机不提供系统栈和与栈操作相关的指令 (包括函数调用等). 你可以手动实现 (如果资源够的话), 或者不用.

下面介绍 Z86+ 程序的语法.

一个合法的源操作数 (`source`) 是以下四者之一:

1. 立即寻址: 十进制 (例如 `9`, `0`, `-1`) 或十六进制 (例如 `0xff`, `-0X3F`). 必须是一个 32 位带符号整数.
2. 寄存器寻址: 例如 `r1`, `r3`.
3. 直接寻址: 例如 `a[0]`, `n`.
4. 寄存器间接寻址: 例如 `a[r2]`.

所有数据类型是 32 位带符号整数. 我们没有提供显式类型转换机制.

目标操作数 (`dest`) 通常是赋值语句等号左边的数. 除了直接寻址不能用外, 与原操作数的类型相同.

一个合法的语句写在一行内, 具有以下形式:

```
[label:] statement [;comment]
```

`label` 以英文和数字组成但必须以英文字母开头, 可有可无, 其唯一用途就是用于 `goto` 的跳转. 为避免歧义请尽可能避免使用含有 if, goto, halt 的词汇作为 label. 与汇编语言不同的是, Z86+ 规定在 label 后面必须紧跟一个语句. 不支持一行单独一个 label.

[2]所谓的 "三地址" 指的就是两个运算分量（操作数 1、操作数 2）及目标操作数三个对象的地址.
[3]1 个字 4 个字节.

注释以 ; 开头. 不支持多行注释. 不支持单独一行只有注释.

Z86+ **对大小写敏感**, 但对空白字符不敏感. 空白符可以是空格或者 tab. 你可以在单词之间、行前、行末添加任意多的空白符, 也可以省去除了换行符、下面特别说明的之外的所有空白符. 只有空白符的行、空行会被直接忽略.

一个合法的 statement 是下面的某一种:

1. 双目赋值语句 `dest := source1 op source2`. 其中 `op` 可以是下面的运算符:
   - 加减乘除模: `+`, `-`, `*`, `/`, `%`.
   - 位左移、位右移: `<<`, `>>`. **注意, 下面所有位操作都是在无符号的语义下进行的**, 例如 `0xffffffff >> 4 = 0x0fffffff`.
   - 循环左移、循环右移: `<<<`, `>>>`. 例如 `0xff00011 <<< 2 = 0x00011ff`.
   - 按位与、按位或、按位异或: `&`, `|`, `^`.
   - 小于、小于等于、大于、大于等于: `<`, `<=`, `>`, `>=`.
   - 等于、不等于: `==`, `!=`.
   - 逻辑与、逻辑或: `&&`, `||`.

   注: 你或许会感到意外 Z86+ 没有提供逻辑非运算. 但事实上逻辑非运算是完全没有必要的.
2. 单目赋值语句 `dest := op source`. 其中 `op` 可以是:
   - 取相反数: `-`.
   - 按位取反: `~`.
3. 赋值语句 `dest := source`.
4. 无条件转移语句 `goto label`. `goto` 后面的空白符是强制的.
5. 条件转移语句 `if source goto label`. `if`, `source`, `goto` 后面必须加空白符.
6. 条件转移语句 `if source1 op source2 goto label`. 转移条件是两个操作数操作后的结果非 0. `if`, `source2`, `goto` 后面必须加空白符.
7. 停机语句 `halt`. 不需要显式停机, 如果程序运行到最后没有更多的语句可以执行, 程序会自动退出.

感谢你一直读到这里.

输入

输入四个整数 n_{min}, n_{max}, A, M. $[n_{min}, n_{max}]$ 是可能出现的 n 的范围, A 是提供的寄存器数量, M 是内存区域有多少个字 (存储 n 的那个字不算在内).

输出

由于 OJ 不支持对 Z86+ 语言进行直接评测, 要求输出一段 Z86+ 的代码. 代码长度应不超过 10^4 行 (包括空行), 代码字节数应不超过 65536 字节.

评分

为了方便考生调试, 并对上文中可能尚未解释清楚的部分进行查询, Z86+ 解释器可以在本题提交界面上下载. 该解释器使用 C++ 编写, **请使用 C++11 进行编译**. 评测系统使用相同的解释器执行你输出的代码.

并使用**按照一定算法构造的**不同的测试数据 (总共约 20 组) 运行多次. 如果程序均在运行 10^7 条语句前返回了正确结果, 则认为该程序通过了该子任务, 获得相应分数.

子任务 1 (10 分)

$n_{min} = n_{max} = 4$, $A = 32$, $M = 16$.

子任务 2 (10 分)

必须先通过子任务 1.

$n_{min} = 2$, $n_{max} = 700$, $A = 8$, $M = 1500$.

子任务 3 (18 分)

必须先通过子任务 2.

$n_{min} = 2$, $n_{max} = 700$, $A = 4$, $M = 700$.

子任务 4 (10 分)

不依赖前面的子任务情况.

$n_{min} = n_{max} = 2^{14}$, $A = 16$, $M = 2^{17}$.

子任务 5 (19 分)

先通过子任务 4. $n_{min} = n_{max} = 2^{16}$, $A = 8$, $M = 2^{17}$.

子任务 6 (8 分)

$n_{min} = 2$, $n_{max} = M = 10\ 000$, $A = 8$.

子任务 7 (7 分)

先通过子任务 6. $n_{min} = 10\ 001$, $n_{max} = M = 20\ 000$, $A = 7$.

子任务 8 (6 分)

先通过子任务 7. $n_{min} = 20\ 001$, $n_{max} = M = 40\ 000$, $A = 6$.

子任务 9 (6 分)

先通过子任务 8. $n_{min} = 40\ 001$, $n_{max} = M = 70\ 000$, $A = 5$.

子任务 10 (6 分)

先通过子任务 9. $n_{min} = 70\ 001$, $n_{max} = M = 100\ 000$, $A = 4$.

样例

standard input	standard output

3 3 32 16	`    if a[0] > a[1] goto L1`
	`    goto L2`
	`L1: r1 := a[0]`
	`    a[0] := a[1]`
	`    a[1] := r1`
	
	`L2: if a[0] > a[2] goto L3`
	`    goto L4`
	`L3: r2 := a[0]`
	`    a[0] := a[2]`
	`    a[2] := r2`
	
	`L4: if a[1] > a[2] goto L5`
	`    goto L6`
	`L5: r1 := a[1]`
	`    a[1] := a[2]`
	`    a[2] := r1`
	
	`L6: halt`

样例提供了一种 "3 个数排序" 的可能的输出. 遗憾的是, 这种输出并不能通过任何一个子任务.

提示

本题鼓励你判断不同的输入来决定采用不同的策略. 但优秀的解法应该具有普适性, 可以轻松通过所有子任务.

解释器**在本题提交界面上下载**. 请使用 C++11 编译. 假设解释器被编译成 `sim` 文件.

- 在 Windows 命令行下: `sim.exe program.zs data.txt <A> <M>`. 其中 `program.zs` 是 Z86+ 代码文件, `data.txt` 是待排序的数据 (直接把所有数按照顺序写在文件里就可以, 会全部读完自动判断数组长度). `A` 和 `M` 是可选参数, 表示机器的配置.
- 在 Linux 下:
  - 编译: `g++ -o sim sim.cpp`
  - 运行: `./sim program.zs data.txt <A> <M>`.

给出的解释器默认运行在 Debug 模式下, Debug 模式下的解释器会输出词法语法分析、运行时等各阶段的详细信息.

如果你想要打印整段文本的话可以尝试使用 Python 3 下的 `print("""text""")`.