Alice、Bob 与 Circuit (abc)

Cyberland 电路学会总共有 n 名成员，每一名成员都有一个自己喜欢的数字，和一个与其他人都互不相同的名字。（不保证每个人喜欢的数字互不相同）

这些成员之间总共互相寄出了 m 封信件，每一封信件都包含了一个发送者和一个接收者。信件的内容恰好为发送者最喜欢的数字。

在所有信件都寄出后，每一名成员会根据他们收到的信件内容(发送者喜欢的数字)，求出这些数字的和对 $65536(2^{16})$ 取模后的值，作为他们的最终存储的数字。

你的任务是确定每个人最终存储的数字。

然而，情况并不像看起来那么简单。Alice, Bob 和Circuit 打算以一种稍微复杂的方式来解决这个问题。具体的：

- Alice 知晓所有的 n 名成员的信息(成员姓名，与其最喜欢的数字)，但是 Alice 不知道关于 m 封信的任何信息。她需要向 Circuit 发送一个长度不超过 10^5 的二进制字符串。
- Bob 知晓所有的 m 封信件信息(信件发送者与接收者的名字)，但是 Bob 不知道关于 n 名成员的信息。他需要向 Circuit 发送一个长度不超过 10^5 的二进制字符串。
- Circuit 是一个电路，其可以接受来自于 Alice 和 Bob 发送的二进制字符串，然后生成一个包含了 $16n$ 位二进制字符串作为其输出。但是由于其计算能力是有限的，Circuit 只能够进行基础的逻辑运算(例如:AND, OR, NOT)。

下面我们将详细介绍电路的工作原理。

电路工作细节

门是组成电路的基本元件。一个门包含零个或者两个布尔信号作为输入，一个布尔信号作为输出。电路中总共有两种类型的门：输入门与计算门。

- 输入门没有输入布尔信号，用于表示 Alice 和 Bob 输入的二进制字符串中的某一位。
  - 总共有 $l_A + l_B$ 个输入门，依次编号为 0 至 $l_A + l_B - 1$。其中 l_A, l_B 分别代表 Alice 和 Bob 给出的二进制字符串的长度。
  - 对于任意 $0 \le i < l_A$，第 i 个门的输出是 Alice 给出的二进制字符串的第 i 位；
  - 对于任意 $0 \le i < l_B$，第 $i + l_A$ 个门的输出是 Bob 给出的二进制字符串的第 i 位；
- 计算门有两个输入，用于计算过程。
  - 计算门的编号从 $l_A + l_B$ 开始。
  - 对于每一个计算门，你需要提供两个相关的门编号，代表该计算门的布尔信号输入；以及数字 $p(0 \le p \le 15)$，代表该计算门的类型。

- 为了避免循环依赖的问题，作为输入的两个门的编号必须小于该计算门的编号。
- 假定两个相关门的输出分别为为 x_0, x_1，则该计算门的输出为：

$$f(p, x_0, x_1) = \left\lfloor \frac{p}{2^{x_0+2x_1}} \right\rfloor \bmod 2$$

以下是一些可能对你有用的示例：

x_0	x_1	x_0 AND x_1 $f(8, x_0, x_1)$	x_0 OR x_1 $f(14, x_0, x_1)$	x_0 XOR x_1 $f(6, x_0, x_1)$	NOT x_0 $f(5, x_0, x_1)$
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

实现细节

请注意：

- 所有数组下标都从 0 开始。 例如，如果 `a` 是一个长度为 `n` 的数组，那么 `a[0]` 到 `a[n-1]` 都是该数组有效数据，访问超出该范围的索引可能会导致越界错误 。
- 所有字符串都以空字符 `\0` 结尾。

你应该实现以下过程：

Alice

```
int alice(const int n, const char names[][5],
          const unsigned short numbers[], bool outputs_alice[]);
```

类型	值	长度	含义	限制
输入	n	1	n	$0 \le n \le 700$
	names	n	每一名成员的名字。	所有成员的名字互不相同。名字仅包含小写英文字母，且长度不超过 4。
	numbers	n	每名成员喜欢的数字。	喜欢的数字为 0 到 65535 的整数。
输出	outputs_alice	l_A	发送给 Circuit 的二进制字符串。	

类型	值	长度	含义	限制
	(返回值)	1	l_A	你需要确保 l_A 不超过 10^5。并且当 n 相同时，l_A 必须相同。

Bob

```
int bob(const int m, const char senders[][5],
        const char recipients[][5], bool outputs_bob[]);
```

类型	值	长度	含义	限制
输入	m	1	m	$0 \le m \le 1000$
	senders	m	信件发送者的名字。	所有名字必定出现在 Alice 的输入中。
	recipients	m	信件接收者的名字。	
输出	outputs_bob	l_B	发送给 Circuit 的二进制字符串。	
	(返回值)	1	l_B	你需要确保 l_B 不超过 10^5。并且当 m 相同时，l_B 必须相同。

Circuit

为了保证 Circuit 的计算过程和一般的电路一样，你无法直接获取 Alice 和 Bob 发送给 Circuit 的二进制串。你只需要知道这两个字符串的长度，并根据字符串长度输出电路结构。

```
int circuit(const int la, const int lb, int operations[],
            int operands[][2], int outputs_circuit[][16]);
```

类型	值	长度	含义	限制
输入	la	1	l_A	
	lb	1	l_B	
输出	operations	l	电路中每个门执行的操作类型。	每个元素为 0 至 15 的整数。

类型	值	长度	含义	限制
	operands	l	生成该计算门的布尔信号输入的计算门编号。	输入计算门编号必然小于当前计算门编号。
	outputs_circuit	n	电路输出门的编号。	`outputs_circuit[i][j]` 表示第 i 名成员最终存储的数字的第 j 位(从最低位开始计算)。成员按照 Alice 输入的顺序编号。
	(返回值)	1	l, 用于表示电路中使用的门的数量(包括输入门)。	你需要保证 $l \leq 2 \times 10^7$。

虽然你可以在 `operations` 和 `operands` 数组中修改编号小于 $l_A + l_B$ 的门的信息，但交互库会忽略这种修改。

例子

考虑如下样例调用序列:

```
alice(3, {"alic", "bob", "circ"}, {10000, 20000, 30000}, outputs_alice);
bob(5, {"alic", "bob", "bob", "circ", "circ"},
    {"circ", "circ", "alic", "circ", "circ"}, outputs_bob);
```

该样例代表了以下场景:

- Alice 知道有 3 个成员，名为 `alic` 的成员最喜欢的数字是 10000，等等。 `alice()` 的可能输出是:
  - `alice()` 的返回值为 2，表示 $l_A = 2$。
  - 在 `alice()` 函数中，设置 `outputs_alice[0] = 1`, `outputs_alice[1] = 0`，表示 Alice 返回的二进制字符串为 10。
- Bob 知道有 5 封信，第一封信是从 `alic` 送出，`circ` 接受等。`bob()` 的可能输出是:
  - `bob()` 的返回值为 3，表示 $l_B = 3$。
  - 在 `bob()` 函数中，设置 `outputs_bob[0] = 1`, `outputs_bob[1] = 1`, `outputs_bob[2] = 0`，表示 Bob 返回的二进制字符串为 110。

根据 `alice()` 和 `bob()` 先前输出的信息，交互库将有如下调用:

```
circuit(2, 3, operations, operands, outputs_circuit);
```

`circuit()` 的一种正确输出是

- `circuit()` 的返回值为 7，这意味着我们添加了两个计算门，标记为 5 和 6。

- 在 `circuit()` 中，按以下方式设置 `operations`、`operands` 和 `outputs_circuit` 数组：
  - `operations = {-1, -1, -1, -1, -1, 8, 14}`。这里我们使用 `-1` 表示被忽略的来自输入门的信息；
  - 操作数 `= {{-1, -1}, {-1, -1}, {-1, -1}, {-1, -1}, {-1, -1}, {0, 4} , {2, 5}}`；
  - `outputs_circuit = {{5, 5, 5, 5, 5, 6, 5, 5, 5, 6, 6, 6, 5, 5, 6, 5}, ...}`。 数组长度较长，完整数组可以查看附件中的`abc.cpp`。

根据 `circuit()` 输出，Circuit 的计算过程如下:

- 添加类型为 8 的计算门，编号为 5，其输入来自输入门 0 和输入门 4。 门 0 的输出是来自 Alice 的字符串的第 0 位，即 1； 门 4 的输出是来自 Bob 的字符串的第 2 位，即 0。 所以计算门 5 的输出是 $f(8,0,1) = 0 \text{ AND } 1 = 0$；
- 添加类型为 14 的计算门，编号为 6，输入来自输入门 2 和计算门 5。 门 2 的输出是来自 Bob 的字符串的第 0 位，即 1； 门 5 的输出是 0。 所以计算门 6 的输出是 $f(14,1,0) = 1 \text{ OR } 0 = 1$；
- `output_circuit[0]` 表示 `alic` 的最终存储的数字，即 $(0100111000100000)_2 = 20000$。 由于 alic 只收到了来自 bob 的一封信，所以 alic 的最终结果是 20000；
- `bob` 的最终存储的数字应该是 0，因为他没有收到信件； `circ` 的最终存储的数字应该是 $(10000 + 20000 + 30000 + 30000) \bmod 65536 = 24464$ 。

附件中的 `abc.cpp` 可以通过这个样例，但我们不保证它可以通过其他测试用例。

约束条件

对于所有测试用例：

- $0 \le n \le 700, 0 \le m \le 1000$。
- 所有成员的名称互不相同，最大长度为 4 个字符，且仅由小写英文字母组成。
- 每个成员最喜欢的数字为 0 到 65535 之间的整数。
- 所有发件人和收件人的姓名都出现在 Alice 的输入数组 `names` 中。
- `alice()` 和 `bob()` 的内存限制均为 2048 MiB ，时间限制均为 0.02 秒。
- `circuit()` 的内存限制为 2048 MiB，时间限制为 7 秒。

在最终测试中，**`alice()`** 和 **`bob()`** 可能会在单个测试用例中被多次调用。 每次调用的时间限制为 0.02 秒。

子任务

子任务类型 A (12 分)

子任务 1,2,3 属于类型 A。类型 A 保证：

- $n = 1$

每个子任务有着如下的附加限制。

- 子任务 1 (4 分)：$m = 0$。
- 子任务 2 (4 分)：$0 \le m \le 1$。
- 子任务 3 (4 分)：$0 \le m \le 1000$。

子任务类型 B (54 分)

子任务 4,5,6 属于类型 B。类型 B 保证：

- $0 \le n \le 30, \frac{n}{2} \le m \le n^2$
- 不存在两封信，他们拥有相同的发送者与接收者。
- 所有成员的名字都出现在了 Bob 的输入中(也就是，每名成员要么至少发送了一封信，要么至少接受了一封信)。

每个子任务有着如下的附加限制。

- 子任务 4 (24 分)：$n = 26$, 所有成员都名字均为单个英文字符。且在 Alice 的输入中，成员名字的循序为 `a` 至 `z`。
- 子任务 5 (24 分)：$n = 26$。
- 子任务 6 (6 分)：没有额外限制。

子任务类型 C (34 分)

子任务 7,8,9 属于类型 C。类型 C 保证:

- $0 \le n \le 700, 0 \le m \le 1000$。

每个子任务有着如下的附加限制。

- 子任务 7 (18 points)：$n = 676$, 所有成员都名字均为恰好两个个英文字符。且在 Alice 的输入中，成员名字的循序按照字典序从小到大排列(换而言之，按照 `aa`, `ab`, `ac`, ..., `az`, `ba`, ..., `bz`, `ca`, ..., `zz` 的顺序排列)。
- 子任务 8 (10 分)：$n = 676$。
- 子任务 9 (6 分)：没有额外限制。

评测程序示例

评测程序示例按以下格式读取输入：

- 第 1 行：$n\ m$
- 第 $2 + i$ 行 $(0 \le i \le n - 1)$：$names_i\ numbers_i$
- 第 $2 + n + i$ 行 $(0 \le i \le m - 1)$：$senders_i\ recipients_i$.

评测程序示例按照如下格式打印你的答案：

- 如果程序成功结束，样例交互器将输出 n 行，每行包含一个整数，代表你的电路计算的每个成员最终存储的数字。
- 否则，样例交互器不会向标准输出输出任何内容，并将错误消息打印到目录中的文件 `abc.log`。

- 此外，样例交互器会将 l_A, l_B, l 的值和每个函数的运行时间输出到 `abc.log`。

评测程序示例不会检查内存限制，也不会检查在 n / m 相同时，l_A / l_B 必须相同的限制。