愛麗絲，鮑勃和電路 (abc)

Cyberland 電路基金會由 n 個成員組成。每個成員都有他/她喜歡的數字和一個唯一的名字（喜歡的數字可能不同）。

成員之間互相發送了 m 封信。每封信都有發件人和收件人，信的內容是發件人喜歡的數字。

每個成員計算他/她收到的內容（發件人喜歡的數字）的總和，並將其對 65536（即 2^{16}）取模作為他/她的結果數字。

你的任務是確定所有的結果數字。

但是，情況並不像表面看起來的那樣簡單。愛麗絲，鮑勃和電路決定以稍微複雜的方式解決這個問題：

- 愛麗絲知道所有的 n 個成員（名字和喜歡的數字），但不知道有關信件的任何信息。她需要向電路發送一個二進制字符串，長度不超過 10^5。
- 鮑勃知道所有的 m 封信（發件人和收件人的名字），但不知道有關成員的任何信息。他需要向電路發送一個二進制字符串，長度不超過 10^5。
- 電路可以接收愛麗絲和鮑勃發送的二進制字符串，並生成一個由 $16n$ 位組成的二進制字符串作為輸出。由於其計算能力有限，電路只能執行基本的邏輯操作（例如 AND、OR、NOT）。

以下將詳細介紹電路的工作原理。

電路細節

閘是電路的基本元素。閘由零個或兩個布爾輸入和一個布爾輸出組成。有兩種類型的閘：輸入閘和計算閘。

- 輸入閘沒有輸入，表示來自愛麗絲和鮑勃發送的二進制字符串中的位。
- 將有 $l_A + l_B$ 個輸入閘，從 0 到 $(l_A + l_B - 1)$ 進行標記，其中 l_A、l_B 分別是來自愛麗絲和鮑勃的字符串的長度。
- 對於 $0 \le i < l_A$，第 i 個閘的輸出是來自愛麗絲字符串的第 i 個位；
- 對於 $0 \le i < l_B$，第 $(i + l_A)$ 個閘的輸出是來自鮑勃字符串的第 i 個位。
- 計算閘有兩個輸入，表示計算過程。
- 計算閘的標籤從 $(l_A + l_B)$ 開始。
- 對於每個計算閘，您應該提供兩個相依閘的標籤為輸入，以及操作類型 $p(0 \le p \le 15)$。
- 為了防止循環依賴，兩個相依閘的標籤必須小於計算閘的標籤。
- 如果其兩個相依閘的輸出分別為 x_0 和 x_1（$x_0, x_1 \in \{0, 1\}$），則計算閘的輸出為：

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

以下是一些對您有用的示例：

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

實現細節

請注意：

- 所有數組索引從0開始。例如，如果`a`是一個長度為`n`的數組，則`a[0]`到`a[n-1]`是有效數據，訪問超出該範圍的索引可能會導致越界錯誤。
- 所有字符串都以空字符`\0`結尾。

你應該實現以下程序：

Alice (愛麗絲)

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

方向	值	長度	含義	約束
輸入	`n`	1	n	$0 \le n \le 700$
	`names`	n	每個成員的名字。	所有名字都是由小寫英文字母組成的，且最大長度為4個字符。
	`numbers`	n	每個成員的喜愛數字。	每個數字都在0到65535的範圍內。
輸出	`outputs_alice`	l_A	二進制字符串發送到電路。	
	返回值	1	l_A	**你需要確保l_A不超過10^5，並且當n相同時，l_A必須固定。**

Bob (鮑勃)

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

方向	值	長度	含義	約束
輸入	`m`	1	m	$0 \le m \le 1000$
	`senders`	m	每封信件上寄件人的名字。	所有名字都出現在Alice的輸入中
	`recipients`	m	每封信件上收件人的名字。	
輸出	`outputs_bob`	l_B	二進制字符串發送到電路。	
	返回值	1	l_B	**你需要確保l_B不超過10^5，並且當m相同時，l_B必須固定。**

電路

為了確保電路的計算過程就像一般的電路一樣，你不能直接從 Alice 和 Bob 傳送的二進位字符串獲取信息。你只知道這兩個字符串的長度並輸出電路結構。

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

方向	值	長度	意義	限制
輸入	`la`	1	l_A	
	`lb`	1	l_B	
輸出	`operations`	l	電路中每個閘的操作類型。	一個從 0 到 15 的整數。
	`operands`	l	電路中每個閘使用的操作數。	數字必須小於當前閘的標籤。
	`outputs_circuit`	n	電路輸出的閘標籤。	`outputs_circuit[i][j]` 表示第 i 個成員的最終結果的第 j 個位（從最低位開始計數）。成員按照 Alice 的輸入順序排序。
	返回值	1	l，表示閘的總數（包括輸入閘）。	**你需要確保** $l \le 2 \times 10^7$

儘管你可以修改 `operations` 和 `operands` 數組中標籤小於 $l_A + l_B$ 的閘的信息，但是測試程序會忽略此類修改。

範例

考慮以下調用：

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

它表示以下情況：

- Alice 知道有 3 個成員，名為 `alic` 的成員有一個喜歡的數字 10000，等等。 `alice()` 的一個可能的輸出是，
  - `alice()` 的返回值為 2，表示 $l_A = 2$。
  - 在 `alice()` 函數內，設置 `outputs_alice[0] = 1, outputs_alice[1] = 0`，表示結果二進制字符串為 10。
- Bob 知道有 5 個字母，第一個字母是從 `alic` 到 `circ`，等等。`bob()` 的一個可能的輸出是，
  - `bob()` 的返回值為 3，表示 $l_B = 3$。
  - 在 `bob()` 函數內，設置 `outputs_bob[0] = 1, outputs_bob[1] = 1, outputs_bob[2] = 0`，表示結果二進制字符串為 110。

基於 `alice()` 和 `bob()` 的上一個輸出，將會有以下調用：

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

對於這個函數的正確輸出是

- `circuit()` 的返回值為 7，表示我們添加了兩個計算閘，標籤為 5 和 6。
- 在 `circuit()` 內，以以下方式設置 `operations`，`operands` 和 `outputs_circuit`：
- `operations = {-1, -1, -1, -1, -1, 8, 14}`，其中我們使用 `-1` 表示從輸入閘忽略的信息；
- `operands = {{-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}, ...}`。數組有點長，你可以檢查附件中的 `abc.cpp` 以獲取完整數組。

根據輸出，計算過程如下：

- 添加一個類型為 8 的計算閘，其輸入來自閘 0 和閘 4。閘 0 的輸出是 Alice 字符串的第 0 位，即 1；閘 4 的輸出是 Bob 字符串的第 2 位，即 0。因此閘 5 的輸出為 $f(8, 0, 1) = 0 \text{ AND } 1 = 0$。
- 添加一個類型為 14 的計算閘，其輸入來自閘 2 和閘 5。閘 2 的輸出是 Bob 字符串的第 0 位，即 1；閘 5 的輸出是 0。因此閘 6 的輸出為 $f(14, 1, 0) = 1 \text{ OR } 0 = 1$。
- `output_circuit[0]` 表示 `alic` 的最終結果，即 $(0100111000100000)_2 = 20000$。由於 `alic` 只從 `bob` 收到一封信，因此 `alic` 的最終結果為 20000。
- Bob 的最終結果應為 0，因為他沒有收到信；`circ` 的最終結果應為 $(10000 + 20000 + 30000 + 30000) \bmod 65536 = 24464$ 。

附件中的 `abc.cpp` 可以通過此示例，但我們不能保證它可以通過其他測試用例。

約束條件

對於所有的測試案例：

- $0 \le n \le 700, 0 \le m \le 1000$。
- 所有的名稱都是由小寫英文字母構成的，且長度不超過 4 個字元。
- 每位成員的最愛數字範圍在 0 到 65535 之間。
- 所有寄件人和收件人的名稱都會出現在 Alice 的輸入陣列 `names` 中。
- `alice()` 和 `bob()` 的記憶體限制為 2048 MiB，而運行時間限制分別為 0.02 秒。
- `circuit()` 的記憶體限制為 2048 MiB，而運行時間限制為 7 秒。

在最終評估中，可能會在單個測試案例中多次呼叫 `alice()` 和 `bob()`。每次呼叫的時間限制為 0.02 秒。

子任務

A 型子任務（12 分）

子任務 1、2、3 屬於 A 型子任務，其中 $n = 1$。

每個子任務都有以下額外的約束條件：

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

B 型子任務（54 分）

子任務 4、5、6 屬於 B 型子任務，其中：

- $0 \le n \le 30, \frac{n}{2} \le m \le n^2$。

- 沒有兩個字母的寄件人和收件人相同。
- 所有成員的名稱都出現在 Bob 的輸入中（即每位成員都至少發送一封信或接收一封信）。

每個子任務都有以下額外的約束條件：

- 子任務 4（24 分）：$n = 26$，所有成員的名稱都是單個小寫字母，在 Alice 的輸入中，它們按照從 `a` 到 `z` 的順序出現。
- 子任務 5（24 分）：$n = 26$。
- 子任務 6（6 分）：沒有特殊限制。

C 型子任務（34 分）

子任務 7、8、9 屬於 C 型子任務，其中 $0 \le n \le 700, 0 \le m \le 1000$。

每個子任務都有以下額外的約束條件：

- 子任務 7（18 分）：$n = 676$，所有成員的名稱都是兩個小寫字母，在 Alice 的輸入中，它們按照字典序排列（例如，`aa`、`ab`、`ac`，...，`az`、`ba`，...，`bz`、`ca`，...，`zz`）。
- 子任務 8（10 分）：$n = 676$。
- 子任務 9（6 分）：沒有額外的限制。

樣例評分程式

樣例評分程式按以下格式讀取輸入：

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

樣例評分程式按以下格式輸出：

- 如果程式成功完成，樣例評分程式將輸出 n 行，每行包含一個整數，表示您為每位成員實現的函數計算的最終結果。
- 否則，評分員不會輸出任何內容到 stdout，而是將錯誤消息輸出到目錄中的 `abc.log` 文件中。
- 此外，樣例評分程式還會將每個函數的 l_A、l_B、l 值和運行時間輸出到 `abc.log`。

樣例評分程式不會檢查記憶體限制和相同 n / m 的 l_A / l_B 必須相等的限制。