愛麗絲、鮑伯與電路小子 (Alice, Bob, and Circuit)

電馭電路基金會 (Cyberland Circuit Foundation) 有 n 名成員。每一名成員都有一個他/她喜好的數字和一個與眾不同的名字 (喜好的數字可能相同)。

已經有 m 封信件在成員之間被傳遞。每一封信有一名寄件者和一名收件者，且信件的內容為寄件者喜好的數字。

每一名成員計算了他/她所收到的所有信件內容 (寄件者喜好的數字) 的總和，然後把除以 65536 (也就是 2^{16}) 得到的餘數作為他/她的結果數字。

你的任務就是要算出所有的結果數字。

然而，事情絕對沒有你想像中的直接。愛麗絲、鮑伯與電路小子決定用一個更複雜的方法來解決以上的問題。

- **愛麗絲** (Alice) 知道所有的 n 名成員 (包含他們的名字與喜好的數字)，但是她並不知道任何與信件相關的資訊。她需要告訴**電路小子**一個長度不超過 10^5 的二元字串。
- **鮑伯** (Bob) 知道所有的 m 封信件 (包含每一封信的寄件者與收件者)，但是他並不知道任何與寄件者和收件者本身的資訊。他需要告訴**電路小子**一個長度不超過 10^5 的二元字串。
- **電路小子** (Circuit) 收到了來自**愛麗絲**與**鮑伯**的兩個二元字串，並接著產生另一個長度為 $16n$ 個字元的二元字串。不過，由於電路小子計算能力的限制，他只能進行一些較為基本的邏輯運算 (例如 AND, OR, NOT 等等)。

接下來，我們將仔細介紹電路是如何運作的。

電路運作細節 (Circuit Details)

閘 (gate) 是形成電路的基本元件。一個閘包含了零個或兩個布林輸入 (取決於閘的種類) 以及一個布林輸出。閘有分兩種：**輸入閘** (input gate) 以及**運算閘** (computation gate)。

- 輸入閘沒有輸入，且它將代表愛麗絲與鮑伯傳送的二元字串內的位元。
  - 總共有 $l_A + l_B$ 個輸入閘，編號依序為 0 到 $(l_A + l_B - 1)$，其中 l_A, l_B 依序代表愛麗絲與鮑伯送出的字串長度。
  - 對於 $0 \le i < l_A$，編號為 i 的閘將輸出愛麗絲送出的字串中的第 i 個位元；
  - 對於 $0 \le i < l_B$，編號為 $i + l_A$ 的閘將輸出鮑伯送出的字串中第 i 個位元。
- 計算閘包含兩個輸入，且描述了電路小子的計算過程。
  - 計算閘的編號從 $l_A + l_B$ 開始。
  - 對於每一個計算閘，你必須提供兩個依賴的閘門編號、以及一個**操作類別** $p (0 \le p \le 15)$。
    - 為了避免循環依賴的狀況發生，依賴的閘門編號必須要小於該計算閘的編號。
    - 若依賴的閘門之輸出位元分別是 x_0 與 x_1 ($x_0, x_1 \in \{0, 1\}$)，那麼該計算閘的輸出為：

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

以下是一些你可能覺得相當有用的範例：

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

實作細節 (Implementation Details)

請注意：

- 所有陣列的註標編號皆從 0 開始。比如說，若 `a` 是一個長度為 `n` 的陣列，那麼 `a[0]` 與 `a[n-1]` 皆為合法的陣列資料，存取超過範圍的陣列資料可能會導致 out-of-bound 相關的錯誤。
- 所有的字串皆以空字元 `\0` 作結。

你應實作以下函式：

愛麗絲 (Alice)

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

資料方向 (Direction)	參數 (Value)	資料長度 (Length)	意義 (Meaning)	範圍限制 (Constraint)
輸入 Input	n	1	n	$0 \le n \le 700$
	names	n	每位成員的名字	所有名字皆相異、僅包含小寫英文字母且長度至多為 4。
	numbers	n	每位成員喜好的數字	所有數字的範圍在 0 與 65535 之間
輸出 Output	outputs_alice	l_A	傳給電路小子的二元字串	
	回傳值 (Return value)	1	l_A	**你必須保證 l_A 不超過 10^5。此外，當輸入之 n 值相同時，l_A 也必須相同。**

鮑伯 (Bob)

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

資料方向 (Direction)	參數 (Value)	資料長度 (Length)	意義 (Meaning)	範圍限制 (Constraint)
輸入 Input	m	1	m	$0 \le m \le 1000$
	senders	m	每封信件的寄件者名字	所有名字保證出現在愛麗絲手上的名單裡面
	recipients	m	每封信件的收件者名字	
輸出 Output	outputs_bob	l_B	傳給電路小子的二元字串	
	回傳值 (Return value)	1	l_B	**你必須保證 l_B 不超過 10^5。此外，當輸入之 m 值相同時，l_B 也必須相同。**

電路小子 (Circuit)

為了保證電路小子的計算過程與真實電路相同，你無法直接取得愛麗絲與鮑伯送給電路小子的字串。你必須在僅得知兩個字串的長度的情形下，輸出電路的結構。

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

資料方向 (Direction)	參數 (Value)	資料長度 (Length)	意義 (Meaning)	範圍限制 (Constraint)
輸入 Input	la	1	l_A	
	lb	1	l_B	
輸出 Output	operations	l	每一個計算閘的操作類別	一個介於 0 與 15 之間的整數
	operands	l	運算元對應的閘門編號	閘門編號必須小於當前的計算閘編號

資料方向 (Direction)	參數 (Value)	資料長度 (Length)	意義 (Meaning)	範圍限制 (Constraint)
	outputs_circuit	n	作為整個電路最終輸出的閘門編號	`outputs_circuit[i][j]` 代表了第 i 名成員的**結果數字**裡的第 j 位元 (從最低有效位元開始起算)。成員順序需與愛麗絲輸入的成員列表相同。
	回傳值 (Return value)	1	l, 代表了電路中閘門的總數 (包含輸入閘)	**你必須保證** $l \leq 2 \times 10^7$

儘管你可以隨意改變 `operations` 與 `operands` 陣列中編號小於 $l_A + l_B$ 的資料值，評分程式還是會忽略你的修改。

範例 (Example)

考慮以下函式呼叫：

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

它代表了以下的情境：

- 愛麗絲 (Alice) 知道總共有 3 名成員，其中名字為 `alic` 的成員喜好的數字為 10000，以此類推。一個 `alice()` 可能的輸出為：
  - `alice()` 回傳值為 2，代表 $l_A = 2$。
  - 在 `alice()` 函式中，寫入 `outputs_alice[0] = 1` 且 `outputs_alice[1] = 0`，代表輸出的二元字串為 `10`。
- 鮑伯 (Bob) 知道總共有 5 封信，其中第一封信是 `alic` 寄給 `circ` 的，以此類推。一個 `bob()` 可能的輸出為：
  - `bob()` 回傳值為 3，代表 $l_B = 3$。
  - 在 `bob()` 函式中，寫入 `outputs_bob[0] = 1`、`outputs_bob[1] = 1` 且 `outputs_bob[2] = 0`，代表輸出的二元字串為 `110`。

根據前述 `alice()` 與 `bob()` 函式呼叫的輸出，評分程式將進行下列的呼叫：

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

一個會被判定為正確的輸出為：

- `circuit()` 回傳值為 7，代表我們額外增加了兩個計算閘，編號為 5 和 6。
- 在 `circuit()` 函式中給定的 `operations`、`operands` 和 `outputs_circuit` 三個陣列裡寫入以下資料：
  - `operations = {-1, -1, -1, -1, -1, 8, 14}`，其中我們用 `-1` 表示可以被忽略的輸入閘相關資訊；
  - `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}, ...}`。這個陣列有點長，完整的陣列內容請參考附件包內的 `abc.cpp`。

根據以上輸出，電路小子的計算過程如下：

- 增加一個操作類別為 8 的計算閘 (編號為 5)，該計算閘的兩個運算元由編號為 0 與編號為 4 的兩個輸入閘提供。編號為 0 的輸入閘代表了愛麗絲提供的字串的第 0 個位元 (在這個例子中為 `1`)；編號為 4 的輸入閘代表了鮑伯提供的字串的第 2 個位元 (在這個例子中為 `0`)；因此，這個編號為 5 的計算閘的輸出為 $f(8, 0, 1) = 0 \text{ AND } 1 = 0$。
- 增加一個操作類別為 14 的計算閘 (編號為 6)，該計算閘的兩個運算元由編號為 2 與編號為 5 的兩個輸入閘提供。編號為 2 的輸入閘代表了鮑伯提供的字串的第 0 個位元 (在這個例子中為 `1`)；編號為 5 的計算閘輸出為 `0`；因此，這個編號為 6 的計算閘的輸出為 $f(14, 1, 0) = 1 \text{ OR } 0 = 1$。
- `output_circuit[0]` 代表了 `alic` 的結果數字 $(0100111000100000)_2 = 20000$。由於 `alic` 只收到了一封來自 `bob` 的信，因此 `alic` 的結果數字為 20000。
- `bob` 的結果數字應為 0，因為他並沒有收到任何信件。而 `circ` 的結果數字應為：

$$(10000 + 20000 + 30000 + 30000) \bmod 65536 = 24464$$

在附件包中的 `abc.cpp` 可以通過上述範例測試，但是我們無法保證該程式碼可以通過其他測試資料。

限制 (Constraints)

對於所有測試資料：

- $0 \le n \le 700, 0 \le m \le 1000.$
- 所有成員的名字皆不同、僅包含小寫英文字母且長度至多為 4。
- 每一位成員喜好的數字範圍為 0 至 65535 之間。
- 所有寄件者與收件者的名字保證出現在愛麗絲的輸入陣列 `names` 裡面。
- 每一次呼叫 `alice()` 與 `bob()` 函式執行的記憶體限制為 2048 MiB 且時間限制為 0.02 秒。
- 每一次呼叫 `circuit()` 函式執行的記憶體限制為 2048 MiB 且時間限制為 7 秒。

在最終評分時，`alice()` 與 `bob()` 可能會在同一筆測試資料中被呼叫多次。 每一次的呼叫執行時間限制為 0.02 秒。

子任務 (Subtasks)

A 型態子任務 (12 points)

A 型態子任務包含子任務 1, 2, 3。在此型態中保證 $n = 1$。

每一個子任務有額外的範圍限制：

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

B 型態子任務 (54 points)

B 型態子任務包含子任務 4, 5, 6。在此型態中有以下保證：

- $0 \le n \le 30, \frac{n}{2} \le m \le n^2.$
- 所有信件的 (收件者, 寄件者) 對皆不同。
- 所有成員的名字皆出現在鮑伯的輸入中。也就是說，每一名成員至少寄出或收到一封信。

每一個子任務有額外的範圍限制：

- 子任務 4 (24 points): $n = 26$、且所有成員的名字皆為單一小寫英文字母。此外，保證 `a` 到 `z` 依序出現在愛麗絲的輸入中。
- 子任務 5 (24 points): $n = 26$。
- 子任務 6 (6 points): 無其他限制。

C 型態子任務 (34 points)

C 型態子任務包含子任務 7, 8, 9。在此型態中保證了 $0 \le n \le 700, 0 \le m \le 1000$。

每一個子任務有額外的範圍限制：

- 子任務 7 (18 points): $n = 676$、且所有成員的名字皆為兩個小寫英文字母組成。在給愛麗絲的輸入中，所有名字依照字典順序出現：`aa`, `ab`, `ac`, ..., `az`, `ba`, ..., `bz`, `ca`, ..., `zz`。
- 子任務 8 (10 points): $n = 676.$
- 子任務 9 (6 points): 無其他限制。

範例評分程式 (Sample Grader)

範例評分程式依照以下格式讀取輸入：

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

範例評分程式依照以下格式輸出：

- 若你的程式成功結束執行，那麼範例評分程式將輸出 n 列，每一列包含一個整數，代表你實作的函式計算出每一名成員的結果數字。
- 否則，範例評分程式將不會輸出任何訊息到 stdout 中。在此同時，範例評分程式會將錯誤訊息輸出至同一個資料夾底下的 `abc.log` 檔案中。
- 此外，範例評分程式還會在 `abc.log` 檔案中輸出 l_A, l_B, l 以及每一次函數呼叫的執行時間。

範例評分程式不會檢查記憶體限制、也不會檢查對於相同的 n / m 輸入，輸出之 l_A / l_B 必須相同。