AliceとBobとCircuit

サイバーランド回路財団は n 人のメンバーからなる。それぞれのメンバーは好きな値と、互いに相異なる名前を持っている（好きな値は互いに相異なるとは限らない）。

m 通の手紙がメンバーの間で送られた。それぞれの手紙には送り主と受取人がおり、手紙の内容は送り主の好きな値である。

それぞれのメンバーは自分が受け取った手紙に書かれた値（送り主の好きな値）の総和を計算し、それを 65536 (すなわち 2^{16}) で割った余りを求め、それを結果の値とする。

あなたのタスクはすべての結果の値を決定することである。

しかし、状況はそれほど単純ではない。 Alice、Bod は Circuit (回路) を用いて問題をいくらかさらに複雑な方法で解くことにした。

- Alice は n 人のメンバー全員の好きな値と名前を知っているが、手紙については何も知らない。彼女はCircuitに長さ 10^5 以下の二進数の列を送る必要がある。
- Bob は m 通の手紙すべての送り主と受取人の名前を知っているが、メンバーについては何も知らない。彼は Circuit に長さ 10^5 以下の二進数の列を送る必要がある。
- Circuit は Alice と Bob から送られた二進数の列を受け取り、続いて $16n$ ビットの二進数の列を生成し出力する。しかし、計算機の性能の都合上、Circuit は基本的な論理演算のみを行える（例えばAND, OR, NOT)。

以下では、Circuitがどのように動くのかの詳細を説明する。

Circuit(回路)の詳細

ゲートはCircuitの基本的な構成要素である。ゲートはゲートの種類に依存して 0 または 2 個の真理値の入力と 1 個の真理値の出力を行う。入力ゲートと計算ゲートの二つの種類のゲートがある。

- 入力ゲートに対する入力はなく、AliceとBobによって送られた文字列を表す。
  - 0 から $(l_A + l_B - 1)$ までの番号が付いた $l_A + l_B$ 個の入力ゲートがある。ただし l_A, l_B はそれぞれAlice, Bobから送られた文字列の長さである。
  - $0 \le i < l_A$ に対して、i 番目のゲートの出力はAliceから送られた二進数の列の i 番目のビットである。
  - $0 \le i < l_B$ に対して、$(i + l_A)$ 番目のゲートの出力はBobから送られた文字列の i 番目のビットである。
- 計算ゲートは 2 つの入力を受け取り、計算の方法を表す。
  - 計算ゲートの番号は $(l_A + l_B)$ から始まる。
  - それぞれの計算ゲートに対して、入力として用いる二つの従属ゲートの番号と演算の種類 $p (0 \le p \le 15)$ を与える必要がある。
    - サイクル状の従属関係が発生しないようにするため、二つの従属ゲートの番号はその計算ゲートの番号より小さくなければならない。
    - 二つの従属ゲートの出力がそれぞれ x_0, x_1 ($x_0, x_1 \in \{0, 1\}$)であった場合、計算ゲートの出力は

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

となる。

以下の例を参考にしてもよい。

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

実装の詳細

注意:

- すべての配列のインデックスは 0 から始まる. 例えば, もし `a` が長さ `n` の配列だった場合, `a[0]` から `a[n-1]` までが正しいデータであり、この範囲を超えてアクセスしようとすると範囲外エラーとなる可能性がある。
- すべての文字列はヌル終端文字 `\0` で終了する。

あなたは以下の関数を実装する必要がある:

Alice

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

方向	値	長さ	意味	制約
入力	`n`	1	n	$0 \leq n \leq 700$
	`names`	n	各メンバーの名前	すべての名前は相異なり、英小文字のみからなり、長さ 4 以下の文字列である。
	`numbers`	n	各メンバーの好きな値	各値は 0 以上 65535 以下の整数である。
出力	`outputs_alice`	l_A	この二進数の文字列がCircuitに送られる。	
	戻り値	1	l_A	**l_A が 10^5 を超過してはならない。また、 n が同じであるとき l_A は同じ値でなければならない。**

Bob

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

方向	値	長さ	意味	制約
入力	`m`	1	m	$0 \leq m \leq 1000$
	`senders`	m	各手紙の送り主の名前	すべての名前はAliceへの入力に現れる。
	`recipients`	m	各手紙の受取人の名前	
出力	`outputs_bob`	l_B	この二進数の文字列がCirucuitに送られる。	
	返り値	1	l_B	**l_B が 10^5 を超過してはならない。また、 m が同じであるとき、 l_B は同じ値でなければならない。**

Circuit

Circuitでの計算方法が普通の回路と同様であることを保証するため、あなたはAliceとBobからCircuitに送られた二進数の文字列を直接得ることはできない。あなたはこれらの二つの文字列の長さのみが知らされた状態で、回路の構成を出力しなければならない。

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

方向	値	長さ	意味	制約
入力	`la`	1	l_A	
	`lb`	1	l_B	
出力	`operations`	l	回路内の各ゲートで行われる演算の種類	0 以上 15 以下の整数

方向	値	長さ	意味	制約
	operands	l	回路内の各ゲートの被演算子の番号	これらの値はこのゲートの番号より小さくなければならない。
	outputs_circuit	n	回路の出力とするゲートの番号	outputs_circuit[i][j] は、i 番目のメンバーの最終的な結果の、最下位ビットから数えて j 番目のビットを表す。メンバーにはAliceへの入力の順に番号が付けられている。
	返り値	1	ゲートの総数を表す値 l (入力ゲートを含む)。	$l \leq 2 \times 10^7$ **とする必要がある。**

あなたはインデックスが $l_A + l_B$ 未満のゲートの情報を operations や operands 配列で修正することもできるが、採点プログラムはそのような修正は無視する。

入出力例

以下のような呼び出しを考える。

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

これは以下のようなシナリオを示している:

- Alice は 3 人のメンバーがいることや, alic という名前のメンバーの好きな値が 10000 であることなどを知っている。 考えられる alice() の出力としては、
  - alice() が2 を返す。これは $l_A = 2$ を意味する。
  - alice() 関数の中で、 outputs_alice[0] = 1, outputs_alice[1] = 0とする。これは結果の二進数の文字列が 10 となることを意味する。
- Bob は 5 通の手紙があり, はじめの手紙が alic から circ へのものだということなどを知っている。考えられる bob() の出力としては、
  - bob() が 3 を返す。これは $l_B = 3$ を意味する。
  - bob() 関数の中で、outputs_bob[0] = 1, outputs_bob[1] = 1, outputs_bob[2] = 0 とする。これは結果の二進数の文字列が 110 となることを意味する。

alice() と bob() の前の出力に基づくと、 次には以下の呼び出しが行われる:

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

この関数の一つの正しい出力は

- circuit() が 7 を返す。これは、番号 5、6 の計算ゲートを追加することを意味する。
- circuit() 関数の中で、 operations, operands, outputs_circuit を以下の方法で定める:
  - operations = {-1, -1, -1, -1, -1, 8, 14} ただし、 -1 はそのインデックスが入力ゲートに対応するため無視される情報を表す。
  - 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}, ...}. この配列は長いため省略されているが、添付ファイルの abc.cpp で完全な配列を確認することができる。

出力によると、計算の手順は、

- ゲート 0 と ゲート 4 から入力を受け取るタイプ 8 の計算ゲートを追加する。ゲート 0 の出力はAliceから送られた文字列の 0 番目のビットであり、1 である。ゲート 4 の出力はBobから送られた文字列の 2 番目のビットであり、 0 である。したがって、ゲート 5 の出力は $f(8,0,1) = 0 \text{ AND } 1 = 0$ となる。
- ゲート 2 と ゲート 5 から入力を受け取るタイプ 14 の計算ゲートを追加する。ゲート 2 の出力はBobから送られた文字列の 0 番目のビットであり、 1 である。ゲート 5 の出力は 0 である。したがってゲート 6 の出力は $f(14,1,0) = 1 \text{ OR } 0 = 1$ である。
- output_circuit[0] は alic の最終的な結果を表し、$(0100111000100000)_2 = 20000$ である。 alic は bob からの一通の手紙のみを受け取るため、 alic の最終的な結果は 20000 である。
- bob の最終的な結果は 0 である。なぜなら、彼は手紙を受け取っていないためである。
- circ の最終的な結果は $(10000 + 20000 + 30000 + 30000) \bmod 65536 = 24464$ である。

添付ファイルのの abc.cpp はこのサンプルには正解するが、他のテストケースに正解することは保証されない。

制約

すべてのテストケースに対して:

- $0 \le n \le 700, 0 \le m \le 1000$.
- すべての名前は相異なり, 英小文字のみからなり, 長さが 4 以下の文字列である。
- 各メンバーの好きな値は 0 以上 65535 以下である。
- すべての送り主と受取人の名前はAliceへの入力の配列 `names` 内に現れる。
- `alice()` と `bob()` はそれぞれメモリー制限が 2048 MiB であり、時間制限が 0.02 秒である。
- `circuit()` はメモリー制限が 2048 MiB であり、時間制限が 7 秒である。

最終的な評価においては、 `alice()` と `bob()` は一つのテストケースの中で複数回呼び出されることがある。 時間制限の 0.02 秒はそれぞれの呼び出しに対するものである。

小課題

小課題タイプ A (12 点)

小課題 1,2,3 は小課題タイプ A に含まれ, $n = 1$ である。

それぞれの小課題では以下の制約が追加される。

- 小課題 1 (4 点): $m = 0$.
- 小課題 2 (4 点): $0 \le m \le 1$.
- 小課題 3 (4 点): $0 \le m \le 1000$.

小課題 タイプ B (54 点)

小課題 4,5,6 は小課題 タイプ B に含まれ、そこでは:

- $0 \le n \le 30, \frac{n}{2} \le m \le n^2$.
- 送り主と受取人が共に同じであるような二つの手紙が存在しない。
- すべてのメンバーの名前は Bob への入力に現れる。 (すなわち、各メンバーは少なくとも一通の手紙を送ったか、少なくとも一通の手紙を受け取ったか、もしくはその両方である）。

それぞれの小課題では以下の制約が追加される。

- 小課題 4 (24 点): $n = 26$, すべてのメンバーの名前は一文字の英小文字であり、Aliceへの入力ではそれらは `a` から `z` の順に現れる。
- 小課題 5 (24 点): $n = 26$.
- 小課題 6 (6 点): 追加の制約はない。

小課題 タイプ C (34 点)

小課題 7,8,9 は 小課題 タイプ C に含まれ、$0 \le n \le 700, 0 \le m \le 1000$ である。

それぞれの小課題では以下の制約が追加される。

- 小課題 7 (18 点): $n = 676$, すべてのメンバーの名前は二文字の英小文字からなる文字列であり、Aliceへの入力ではそれらは辞書順で現れる。(すなわち、 `aa`, `ab`, `ac`, ..., `az`, `ba`, ..., `bz`, `ca`, ..., `zz`)。
- 小課題 8 (10 点): $n = 676$.
- 小課題 9 (6 点): 追加の制約はない。

採点プログラムのサンプル

採点プログラムのサンプルは以下の形式で入力を受け取る:

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

採点プログラムのサンプルは以下の形式で出力を行う。

- もしプログラムが正常に終了した場合, 採点プログラムのサンプルは n 行出力する。それぞれの行は一つの整数からなり、あなたが実装した関数によって計算された各メンバーの最終的な結果を意味する。
- そうでない場合, 採点プログラムのサンプルは stdout には何も出力せず、ディレクトリ内の `abc.log` にエラーメッセージを出力する。

- 加えて,採点プログラムのサンプルは l_A, l_B, l と各関数の実行時間を `abc.log` に出力する。

採点プログラムのサンプルは、メモリー制限や、同じ n / m に対しては l_A / l_B は等しくなければならないという制約をチェックしないことに注意せよ。