앨리스, 밥, 서킷

사이버랜드 회로 재단은 n명의 회원이 있다. 각 회원마다 좋아하는 수가 있고, 이름이 모두 다르다. (좋아하는 수가 같은 회원들이 있을 수 있다.)

회원끼리 m 통의 편지를 보냈다. 각각의 편지에는 보낸 사람, 받는 사람이 있고, 편지의 내용은 보낸 사람이 좋아하는 수이다.

모든 회원은 자신이 받은 편지에 있는 수의 합을 구하고, 65536 (즉, 2^{16})으로 나눈 나머지를 계산한 결과를 구한다.

당신이 할 일은 모든 결과값을 알아내는 것이다.

그런데, 일은 보기보다 더 꼬여있다. 앨리스, 밥, 서킷 셋은 이 문제를 다음과 같이 조금 더 복잡하게 풀기로 했다.

- 앨리스는 회원 n명 모두에 대한 정보(이름, 좋아하는 수)를 알고 있지만, 편지에 대해서는 전혀 아는 것이 없다. 앨리스는 서킷에게 최대 10^5 길이의 이진 문자열을 보낼 필요가 있다.
- 밥은 m 통의 편지 모두에 대한 정보(보낸 사람, 받는 사람)를 알고 있지만, 회원에 대해서는 전혀 아는 것이 없다. 밥은 서킷에게 최대 10^5 길이의 이진 문자열을 보낼 필요가 있다.
- 서킷은 앨리스와 밥이 보낸 이진 문자열을 받고, $16n$ 길이의 이진 문자열을 출력한다. 그러나, 서킷의 연산 능력에는 한계가 있어서 AND, OR, NOT 등 기본적인 논리 연산만 할 수 있다.

다음에서 어떻게 회로가 구성되는지 설명한다.

회로의 상세 명세

게이트는 회로의 기본 성분이다. 게이트는 0개 또는 2개의 부울 입력과 (게이트의 종류에 따라) 1개의 부울 출력으로 이루어진다. 두 타입의 게이트가 있는데, 입력 게이트와 연산 게이트가 있다.

- 입력 게이트는 입력이 없고, 앨리스와 밥이 보낸 이진 문자열의 각각 비트를 표현한다.
  - $l_A + l_B$개의 입력 게이트가 있고, 0부터 $(l_A + l_B - 1)$까지 정수 레이블로 표현된다. l_A, l_B은 각각 앨리스와 밥이 보낸 문자열의 길이이다.
  - $0 \le i < l_A$,일 때 i번 게이트의 출력은 앨리스가 보낸 문자열의 i번째 비트값이다.
  - $0 \le i < l_B$일 때 $(i + l_A)$번 게이트의 출력은 밥이 보낸 문자열의 i번째 비트 값이다.
- 연산 게이트는 두 개의 입력이 있고, 연산 단계를 표현한다.
  - 각각의 연산 게이트는 $(l_A + l_B)$ 이상인 정수 레이블로 표현된다.
  - 각 연산 게이트마다, 두 입력에 연결되는 게이트의 레이블과 이 게이트가 수행하는 연산 타입 $p (0 \le p \le 15)$를 정해주어여 한다. .
    - 회로가 순환적인 의존을 갖지 않게 하기 위해서, 두 입력에 연결되는 게이트의 레이블은 연산 게이트의 레이블보다 작은 수여야 한다.
    - 두 입력 게이트의 출력이 각각 x_0와 x_1 ($x_0, x_1 \in \{0, 1\}$)일 때, 연산 게이트의 출력은 다음과 같다.

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

다음은 유용하게 쓰일 수 있는 예제들이다.

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

상세 구현

다음에 주의하시오.

- 모든 배열의 인덱스는 0부터 시작한다. 예를 들어, `a`가 길이 `n`인 배열이면, `a[0]`부터 `a[n-1]`는 타당한 데이터지만, 이 범위를 넘어가는 인덱스는 out-of-bounds 에러를 발생하게 한다.
- 모든 문자열은 null character `\0`으로 끝난다.

다음 함수를 구현해야 한다.

앨리스

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

방향	변수	길이	의미	제약조건
입력	n	1	n	$0 \le n \le 700$
	names	n	각 회원의 이름	모든 이름은 서로 다르며, 영어 소문자로만 되어 있고, 최대 길이가 4이다.
	numbers	n	각 회원이 가장 좋아하는 수	각각의 수는 0 이상 65535 이하
Output	outputs_alice	l_A	서킷에 보낸 이진 문자열	
	(Return value)	1	l_A	**l_A가 10^5을 넘으면 안되며, n이 동일하다면 l_A가 항상 같아야 한다.**

밥

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

방향	변수	길이	의미	제약조건
입력	m	1	m	$0 \le m \le 1000$
	senders	m	각각의 편지를 보낸 사람의 이름	모든 이름은 앨리스의 입력에 포함된다
	recipients	m	각각의 편지를 받는 사람의 이름	
Output	outputs_bob	l_B	서킷에 보낸 이진 문자열	
	(Return value)	1	l_B	**l_B가 10^5을 넘어서는 안되며, m이 동일하다면 l_B가 항상 같아야 한다.**

서킷

서킷이 연산하는 과정을 일반적인 회로가 동작하는 것과 비슷하게 하기 위해서, 앨리스와 밥이 보낸 이진 문자열을 직접 접근할 수 없다. 이 두 문자열의 길이만 알고 있다.
출력은 회로의 구조이다.

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

방향	변수	길이	의미	제약조건
입력	la	1	l_A	
	lb	1	l_B	
출력	operations	l	회로에서 각각의 게이트가 수행하는 연산의 종류	0 이상 15 이하인 정수.
	operands	l	각 게이트에 들어오는 피연산자 (입력)	게이트의 레이블보다 작은 수여야 한다.
	outputs_circuit	n	회로 출력의 게이트 레이블	`outputs_circuit[i][j]`는 i번째 회원의 결과값의 j 번째 비트이다 (가장 낮은 자리부터 세었을 때) 회원의 순서는 앨리스의 입력에 나오는 순서대로이다.
	(Return value)	1	(입력 게이트를 포함하여) 모든 게이트의 개수 l	**반드시 $l \le 2 \times 10^7$이어야 한다.**

`operations`과 `operands` 배열에서 $l_A + l_B$보다 작은 인덱스를 갖는 게이트에 대한 정보를 수정할 수 있지만, 그레이더는 이를 무시한다.

예제

다음 함수 호출을 생각해보자.

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

다음과 같은 시나리오가 진행된다.

- 앨리스는 회원이 3명 있고, 이름이 `alic`인 회원이 제일 좋아하는 수는 10000이라는 등의 정보를 알게 된다. `alice()`의 가능한 출력은 다음과 같다.
  - `alice()`의 리턴값은 2이고, $l_A = 2$를 의미한다.
  - `alice()` 함수 내부에서, `outputs_alice[0] = 1, outputs_alice[1] = 0`가 되며, 서킷에 전달되는 문자열이 10라는 뜻이다.
- 밥은 편지가 모두 5통이 있고, 첫번째 편지는 `alic`이 `circ`에게 보낸 것이라는 것 등의 정보를 알게 된다. `bob()`의 가능한 출력은 다음과 같다.
  - `bob()`의 리턴값은 3이고, $l_B = 3$을 의미한다.
  - `bob()` 함수 내부에서, `outputs_bob[0] = 1, outputs_bob[1] = 1, outputs_bob[2] = 0`이 되며, 서킷에 전달되는 문자열이 110라는 뜻이다.

`alice()`와 `bob()`의 출력에 따라, 다음과 같이 함수가 호출될 것이다:

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

이 함수의 정확한 출력은 다음과 같다.

- `circuit()`의 리턴값은 7이고, 레이블이 5와 6인 두 개의 연산 게이트가 있다는 뜻이다.
- `circuit()` 함수 내부에서, `operations`, `operands`, `outputs_circuit`의 내용은 다음과 같이 정해진다.
  - `operations = {-1, -1, -1, -1, -1, 8, 14}`인데, 연산을 하지 않는 입력 게이트를 표현하기 위해서 `-1`을 사용한다.
  - `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}, ...}`. 이 배열은 길기 때문에, 정확한 내용을 알기 위해서는 첨부된 `abc.cpp`를 참고하기 바란다.

이 출력을 가지고 계산 과정은 다음과 같이 진행된다.

- 게이트 0과 4에서 입력을 받는 연산 타입 8인 연산 게이트가 추가된다. 게이트 0의 출력 1은 앨리스의 출력 문자열의 0번째 비트이다. 게이트 4의 출력 0은 밥의 출력 문자열의 2번째 비트이다. 따라서 게이트 5의 출력은 $f(8, 0, 1) = 0 \text{ AND } 1 = 0$이다.
- 게이트 2과 5에서 입력을 받는 연산 타입 14인 연산 게이트가 추가된다. 게이트 2의 출력 1은 밥의 출력 문자열의 0번째 비트이다. 게이트 5의 출력은 0이다. 따라서 게이트 6의 출력은$f(14, 1, 0) = 1 \text{ OR } 0 = 1$이다.
- `output_circuit[0]`은 `alic`의 최종 결과를 나타내며, $(0100111000100000)_2 = 20000$이다. `alic`은 `bob`이 보낸 편지 하나만 받기 때문에, `alic`의 최종 결과는 20000이다.
- `bob`의 최종 결과는 0이어야 하는데, 편지를 하나도 받지 못했기 때문이다. `circ`의 최종 결과는 $(10000 + 20000 + 30000 + 30000) \bmod 65536 = 24464$ 이다.

첨부된 `abc.cpp`는 이 예제를 통과할 수 있지만, 다른 테스트 케이스도 통과할 것인지는 보장할 수 없다.

제약 조건

모든 테스트케이스에서

- $0 \le n \le 700, 0 \le m \le 1000$.
- 모든 이름은 서로 다르며, 영어 소문자로만 되어 있고, 최대 길이가 4이다.
- 각 회원이 좋아하는 수는 0 이상 65535 이하인 정수이다.
- 편지를 보낸 사람, 받는 사람의 이름은 모두 앨리스의 입력 배열 `names`에 포함되어 있다.
- `alice()`, `bob()`은 각각 메모리 제한이 2048 MiB, 시간 제한이 0.02 초이다.
- `circuit()`은 메모리 제한이 2048 MiB, 시간 제한이 7 초이다.

최종 채점에서 테스트 케이스 하나에서도 `alice()`와 `bob()`은 여러 번 호출될 수 있다. 각 호출마다 시간 제한은 0.02초이다.

부분 문제

부분 문제 A (12 점)

부분 문제 1,2,3은 부분 문제 A에 속하고, $n = 1$이다.

각각의 부분문제는 다음과 같은 추가적인 제약 조건이 있다.

- 부분 문제 1 (4 점): $m = 0$.

- 부분 문제 2 (4 점): $0 \leq m \leq 1$.
- 부분 문제 3 (4 점): $0 \leq m \leq 1000$.

부분 문제 B (54 점)

부분 문제 4,5,6 은 부분 문제 B에 속한다.

- $0 \leq n \leq 30, \frac{n}{2} \leq m \leq n^2$.
- 두 통의 편지가 보내는 사람, 받는 사람이 같은 경우는 없다.
- 밥의 입력에는 모든 회원이 이름이 포함된다 (즉, 모든 회원은 적어도 한 통의 편지를 보내거나 적어도 한 통의 편지를 받는다.)

각각의 부분문제는 다음과 같은 추가적인 제약 조건이 있다.

- 부분 문제 4 (24 점): $n = 26$, 모든 회원의 이름은 영어 소문자 한 글자이며, 앨리스의 입력에 `a`부터 `z` 순서로 나온다.
- 부분 문제 5 (24 점): $n = 26$.
- 부분 문제 6 (6 점): 추가적인 제약 조건이 없다.

부분 문제 C (34 점)

부분 문제 7,8,9는 부분 문제 C에 속하고, $0 \leq n \leq 700, 0 \leq m \leq 1000$이다.

각각의 부분문제는 다음과 같은 추가적인 제약 조건이 있다.

- 부분 문제 7 (18 점): $n = 676$, 모든 회원의 이름은 영어 소문자 두 글자이며, 앨리스의 입력에 `aa`, `ab`, `ac`, ..., `az`, `ba`, ..., `bz`, `ca`, ..., `zz`와 같이 사전 순서대로 나온다.
- 부분 문제 8 (10 점): $n = 676$.
- 부분 문제 9 (6 점): 추가적인 제약 조건이 없다.

샘플 그레이더

샘플 그레이더는 다음 양식으로 입력을 읽는다.

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

샘플 그레이더는 다음 양식으로 출력한다.

- 만약 프로그램이 정상적으로 종료한다면, 샘플 그레이더는 총 n 줄을 출력하는데, 각 줄은 정수 하나를 포함한다. 이 정수는 여러분의 프로그램이 계산한 각 회원의 최종 결과를 나타낸다.
- 그렇지 않으면, 그레이더는 표준 출력으로 아무것도 출력하지 않고, 디렉토리 안의 `abc.log` 파일에 에러 메시지를 출력한다.
- 추가적으로, 샘플 그레이더는 l_A, l_B, l 값과 각각의 함수의 실행 시간을 `abc.log`에 출력한다.

샘플 그레이더는 메모리 제한을 확인하지 않고, n / m이 같으면 l_A / l_B도 같아야 한다는 제약 조건을 확인하지 않는다.