Alice, Bob, dan Sirkuit

Cyberland Circuit Foundation terdiri dari n buah anggota. Setiap anggota memiliki nomor favorit dan nama yang unik (nomor-nomor favorit tersebut belum tentu berbeda).

m surat telah dikirim di antara anggota-anggota. Setiap surat memiliki pengirim dan penerima, dan isi dari surat tersebut adalah nomor favorit pengirim.

Setiap anggota menghitung jumlah isi-isi surat tersebut (nomor favorit pengirim) yang mereka terima dan mengambil modulo 65536 (yaitu, 2^{16}) sebagai nomor perolehannya.

Tugas Anda adalah menentukan semua nomor-nomor perolehan.

Namun, situasi ini tidak sesederhana kelihatannya. Alice, Bob, dan Sirkuit memutuskan untuk menyelesaikan masalah ini dengan cara yang sedikit rumit:

- Alice mengetahui semua n anggota (nama dan nomor favorit mereka), tetapi tidak mengetahui informasi mengenai surat-surat. Dia perlu mengirim *string* biner ke Sirkuit dengan panjang tidak lebih dari 10^5.
- Bob mengetahui semua m surat (nama pengirim dan nama penerima), tetapi tidak mengetahui informasi mengenai anggota-anggota. Dia perlu mengirim *string* biner ke Sirkuit dengan panjang tidak lebih dari 10^5.
- Sirkuit dapat menerima *string* biner yang dikirim oleh Alice dan Bob, dan kemudian menghasilkan *string* biner yang terdiri dari $16n$ bit sebagai *output*. Namun, karena daya komputasinya yang terbatas, Sirkuit hanya mampu melakukan operasi logika dasar (misal: AND, OR, NOT).

Berikut ini, kami akan memperkenalkan cara kerja sirkuit secara detil.

Detil Sirkuit

Gate adalah elemen dasar dari sebuah sirkuit. *Gate* terdiri dari nol atau dua masukan *boolean* (tergantung pada jenis *gate*), dan satu keluaran *boolean*. Terdapat dua jenis *gate*: *input gate* dan *computation gate*.

- *Input gate* tidak memiliki masukan dan merepresentasikan bit dari *string* biner yang dikirimkan oleh Alice dan Bob.
  - Akan terdapat $l_A + l_B$ *input gate*, diberi label dari 0 hingga $(l_A + l_B - 1)$, dimana l_A, l_B adalah panjang *string* dari Alice and Bob.
  - Untuk $0 \le i < l_A$, keluaran dari *gate* ke-i adalah bit ke-i *string* dari Alice;
  - Untuk $0 \le i < l_B$, keluaran dari *gate* ke-$(i + l_A)$ adalah bit ke-i *string* dari Bob.
- *Computation gate* memiliki dua masukan dan merepresentasikan proses komputasi.
  - Label untuk *computation gate* dimulai dari $(l_A + l_B)$.
  - Untuk setiap *computation gate*, Anda harus memberikan label dari dua *gate* masukan, dan tipe operasi $p (0 \le p \le 15)$.
    - Untuk mencegah *circular dependencies*, label dari kedua *gate* harus lebih kecil dari label *computation gate*.
    - Jika keluaran dari kedua *gate* adalah x_0 dan x_1 ($x_0, x_1 \in \{0, 1\}$), maka keluaran dari *computation gate* adalah:

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

Berikut adalah beberapa contoh yang mungkin berguna untuk Anda:

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

Detil Implementasi

Harap diperhatikan:

- Semua indeks array dimulai dari 0. Misalnya, jika `a` adalah array dengan panjang `n`, maka `a[0]` hingga `a[n-1]` adalah data yang valid, mengakses indeks di luar rentang tersebut dapat menyebabkan error *out-of-bounds*.
- Semua string diakhiri dengan karakter null `\0`.

Anda harus mengimplementasikan prosedur-prosedur berikut:

Alice

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

Arahan	Nilai	Panjang	Pengartian	Batasan
Masukan	n	1	n	$0 \leq n \leq 700$
	names	n	Nama dari setiap anggota.	Semua nama berbeda, hanya terdiri dari huruf *lowercase*, dan memiliki panjang maksimum 4 karakter.
	numbers	n	Nomor favorit dari setiap anggota.	Setiap nomor berada di dalam rentang dari 0 hingga 65535.
Keluaran	outputs_alice	l_A	String biner yang dikirimkan untuk Sirkuit.	
	(Return value)	1	l_A	**Anda perlu memastikan bahwa l_A tidak melebihi 10^5 dan apabila n sama, maka l_A harus sama.**

Bob

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

Arahan	Nilai	Panjang	Pengartian	Batasan
Masukan	m	1	m	$0 \leq m \leq 1000$
	senders	m	Nama pengirim untuk setiap surat.	Semua nama muncul di masukan Alice.
	recipients	m	Nama penerima untuk setiap surat.	
Keluaran	outputs_bob	l_B	String biner yang dikirimkan untuk Sirkuit.	
	(Return value)	1	l_B	**Anda perlu memastikan bahwa l_B tidak melebihi 10^5 dan apabila m sama, maka l_B harus sama.**

Sirkuit

Untuk memastikan bahwa proses komputasi Sirkuit seperti sirkuit pada umumnya, Anda tidak dapat langsung mendapatkan *string* biner yang dikirim dari Alice dan Bob ke Sirkuit. Anda hanya mengetahui panjang dari dua string ini dan mengeluarkan struktur sirkuit tersebut.

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

Arahan	Nilai	Panjang	Pengartian	Batasan
Masukan	la	1	l_A	
	lb	1	l_B	
Keluaran	operations	l	Tipe operasi yang dilakukan oleh setiap *gate* dalam sirkuit.	Sebuah bilangan bulat dari 0 sampai 15.

Arahan	Nilai	Panjang	Pengartian	Batasan
	operands	l	*Operand* yang digunakan oleh setiap gerbang di sirkuit.	Bilangan harus lebih kecil dari label *gerbang* sekarang.
	outputs_circuit	n	Label *gate* dari keluaran sirkuit.	`outputs_circuit[i][j]` menunjukkan bit ke-j (dihitung dari bit paling tidak signifikan) dari hasil akhir untuk anggota ke-i. Para anggota diurutkan sesuai dengan masukan Alice.
	(Return value)	1	l, yang merepresentasikan jumlah *gate* (termasuk *input gate*).	**Anda perlu memastikan bahwa** $l \leq 2 \times 10^7$

Meskipun Anda dapat memodifikasi informasi gerbang dengan indeks kurang dari $l_A + l_B$ dalam array `operasi` dan `operan`, *grader* akan mengabaikan modifikasi tersebut.

Example

Perhatikan pemanggilan-pemanggilan berikut:

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

Pemanggilan ini merepresentasikan skenario berikut:

- Alice mengetahui terdapat 3 buah anggota, anggota dengan nama `alic` memiliki nomor favorit 10000, dll. Keluaran yang mungkin untuk `alice()` adalah,
  - Pengembalian `alice()` adalah 2, merepresentasikan $l_A = 2$.
  - Di dalam fungsi `alice()`, ubah `outputs_alice[0] = 1, outputs_alice[1] = 0`, merepresentasikan bahwa hasil string biner adalah 10.
- Bob mengetahui terdapat 5 buah surat, surat pertama dikirimkan oleh `alic` untuk `circ`, dll. Keluaran yang mungkin untuk `bob()` adalah,
  - Pengembalian `bob()` adalah 3, merepresentasikan $l_B = 3$.
  - Di dalam fungsi `bob()`, ubah `outputs_bob[0] = 1, outputs_bob[1] = 1, outputs_bob[2] = 0`, merepresentasikan bahwa hasil string biner adalah 110.

Berdasarkan keluara sebelumnya untuk `alice()` dan `bob()`, terdapat pemanggilan berikut:

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

Keluaran yang benar untuk fungsi ini adalah

- Pengembalian `circuit()` adalah 7, artinya kita menambahkan dua *computation gate*, dilabeli 5 dan 6.
- Dalam `circuit()`, ubah `operations`, `operands`, dan `outputs_circuit` dengan cara berikut:
  - `operations = {-1, -1, -1, -1, -1, 8, 14}`, dimana kita menggunakan `-1` untuk merepresentasikan informasi yang diabaikan dari *gate-gate* masukan;
  - `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}, ...}`. Array cukup panjang, Anda dapat memeriksa `abc.cpp` dalam lampiran untuk array penuh.

Berdasarkan hasil keluaran, prosedur perhitungan adalah sebagai berikut,

- Tambahkan *computation gate* tipe 8, dengan masukan dari gerbang 0 dan gerbang 4. Keluaran dari gerbang 0 adalah bit ke-0 dari string dari Alice, yang mana adalah 1; Keluaran dari gerbang 4 adalah bit ke-2 dari string dari Bob, yang mana adalah 0. Jadi keluaran untuk gerbang 5 adalah $f(8, 0, 1) = 0 \text{ AND } 1 = 0$.
- Tambahkan *computation gate* tipe 14, dengan masukan dari gerbang 2 dan gerbang 5. Keluaran dari gerbang 2 adalah bit ke-0 dari string dari Bob, yang mana adalah 1; Keluaran dari gerbang 5 adalah 0. Jadi keluaran untuk gerbang 6 adalah $f(14, 1, 0) = 1 \text{ OR } 0 = 1$.

- `output_circuit[0]` merepresentasikan hasil akhir dari `alic`, yang mana adalah $(0100111000100000)_2 = 20000$. Karena `alic` hanya menerima satu surat dari `bob`, hasil akhir dari `alic` adalah 20000.
- Hasil akhir dari `bob` seharusnya adalah 0, karena dia tidak menerima surat apa pun; Hasil akhir dari `circ` seharusnya adalah $(10000 + 20000 + 30000 + 30000) \bmod 65536 = 24464$.

`abc.cpp` dalam lampiran dapat melewati contoh ini, namun kami tidak menjamin bahwa `abc.cpp` dapat melewati kasus uji lainnya.

Batasan

Untuk semua kasus uji:

- $0 \le n \le 700, 0 \le m \le 1000$.
- Semua nama berbeda, hanya terdiri dari huruf lowercase, dan memiliki panjang maksimal 4 karakter.
- Nomor favorit setiap anggota berada dalam rentang 0 hingga 65535.
- Nama semua pengirim dan penerima muncul dalam array masukan `names` milik Alice.
- `alice()` dan `bob()` masing-masing memiliki batas memori sebesar 2048 MiB dan batas waktu sebesar 0.02 detik.
- `circuit()` memiliki batas memori sebesar 2048 MiB dan batas waktu sebesar 7 detik.

Untuk evaluasi akhir, `alice()` dan `bob()` bisa dipanggil beberapa kali dalam satu kasus uji. Batas waktu sebesar 0.02 detik berlaku untuk setiap panggilan.

Subsoal

Subsoal Tipe A (12 poin)

Subsoal 1,2,3 berada dalam subsoal tipe A, dimana $n = 1$.

Setiap subsoal memiliki batasan tambahan sebagai berikut:

- Subsoal 1 (4 poin): $m = 0$.
- Subsoal 2 (4 poin): $0 \le m \le 1$.
- Subsoal 3 (4 poin): $0 \le m \le 1000$.

Subsoal Tipe B (54 poin)

Subtask 4,5,6 berada dalam subsoal tipe B, dimana:

- $0 \le n \le 30, \frac{n}{2} \le m \le n^2$.
- Tidak ada dua surat dengan pengirim dan penerima yang sama.
- Nama semua anggota muncul dalam masukan Bob (yaitu, setiap anggota mengirim setidaknya satu surat atau menerima setidaknya satu surat).

Setiap subsoal memiliki batasan tambahan sebagai berikut:

- Subsoal 4 (24 poin): $n = 26$, semua nama anggota adalah sebuah huruf *lowercase*, dan dalam masukan Alice, mereka muncul berurutan dari `a` hingga `z`.
- Subsoal 5 (24 poin): $n = 26$.
- Subsoal 6 (6 poin): Tidak ada batasan tambahan.

Subsoal Tipe C (34 poin)

Subsoal 7,8,9 berada dalam subsoal tipe C, dimana $0 \le n \le 700, 0 \le m \le 1000$.

Setiap subsoal memiliki batasan tambahan sebagai berikut:

- Subsoal 7 (18 poin): $n = 676$, semua nama anggota adalah dua buah huruf *lowercase*, dan dalam masukan Alice, mereka muncul dalam urutan leksikografis (misalnya, `aa`, `ab`, `ac`, ..., `az`, `ba`, ..., `bz`, `ca`, ..., `zz`).
- Subsoal 8 (10 poin): $n = 676$.
- Subsoal 9 (6 poin): Tidak ada batasan tambahan.

Contoh Grader

Contoh *grader* yang diberikan akan membaca masukan dengan format sebagai berikut:

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

Contoh *grader* akan mencetak jawaban dengan format sebagai berikut:

- Jika program selesai dengan sukses, *grader* contoh akan mencetak n buah baris, masing-masing berisi sebuah bilangan bulat, yang merepresentasikan hasil akhir yang dihitung oleh fungsi-fungsi yang Anda implementasikan untuk setiap anggota.
- Jika tidak, *grader* tidak akan mencetak apa pun ke stdout dan mencetak pesan kesalahan ke file `abc.log` di direktori.
- Selain itu, grader contoh akan mencetak nilai l_A, l_B, l dan waktu eksekusi setiap fungsi ke `abc.log`.

Grader contoh tidak akan memeriksa batas memori dan batasan bahwa untuk n / m yang sama, l_A / l_B harus sama.