อลิซ บ๊อบ และ เซอร์กิต

มูลนิธิวงจรดินแดนไซเบอร์ (Cyberland Circuit Foundation) มีสมาชิกอยู่ n คน สมาชิกแต่ละคนมีชื่อที่แตกต่างกัน และมีตัวเลขที่ชอบอยู่คนละหนึ่งตัว (ตัวเลขที่ชอบของแต่ละคนอาจจะซ้ำกันก็เป็นได้)

มีการส่งจดหมาย m ฉบับไปมาระหว่างสมาชิกเหล่านี้ จดหมายแต่ละฉบับจะมีผู้ส่งและผู้รับ และเนื้อหาของจดหมายคือตัวเลขที่ชอบของผู้ส่ง

สมาชิกแต่ละคนจะคำนวณผลรวมของเนื้อหาของจดหมายที่ได้รับทั้งหมด (ซึ่งคือตัวเลขที่ชอบของผู้ส่ง) และคำนวณเศษของการหารจำนวนดังกล่าวด้วย 65536 (ซึ่งคือ 2^{16}) เป็นผลลัพธ์ของสมาชิกคนดังกล่าว

งานของคุณคือการคำนวณตัวเลขผลลัพธ์ทั้งหมด

อย่างไรก็ตาม สถานการณ์มันไม่ได้ง่ายเช่นนั้น อลิซ บ๊อบ และ เซอร์กิต ได้ตัดสินใจที่จะแก้ปัญหานี้ด้วยวิธีการที่ยุ่งขึ้นเล็กน้อย ดังนี้

- อลิซ รู้ข้อมูลของสมาชิกทั้ง n คน (ทั้งชื่อและตัวเลขที่ชอบ) แต่ไม่รู้ข้อมูลใด ๆ ของจดหมายเลย อลิซจะต้องส่งสายอักขระตัวเลขฐานสองความยาวไม่เกิน 10^5 ไปให้เซอร์กิต
- บ๊อบ รู้ข้อมูลของจดหมายทั้ง m ฉบับ (ชื่อของผู้ส่งและผู้รับ) แต่ไม่รู้ข้อมูลของสมาชิกเลย บ๊อบจะต้องส่งสายอักขระตัวเลขฐานสองความยาวไม่เกิน 10^5 ไปให้เซอร์กิต
- เซอร์กิต สามารถรับสายอักขระตัวเลขฐานสองที่ส่งมาจากอลิซและบ๊อบได้ และสามารถสร้างสายอักขระตัวเลขฐานสองความยาว $16n$ เป็นข้อมูลส่งออก อย่างไรก็ตาม ด้วยข้อจำกัดทางด้านพลังการคำนวณ เซอร์กิตสามารถคำนวณได้แค่ตัวดำเนินการพื้นฐานทางด้านตรรกศาสตร์เท่านั้น (ซึ่งคือ AND, OR และ NOT)

หัวข้อต่อไปนี้อธิบายถึงวิธีการทำงานของเซอร์กิต

รายละเอียดของเซอร์กิต

เกต (Gate) คือองค์ประกอบพื้นฐานของเซอร์กิต เกตประกอบด้วยข้อมูลนำเข้าแบบบูลีน (boolean) 0 หรือ 2 ตัว (ขึ้นอยู่กับประเภทของเกต) และมีข้อมูลส่งออกแบบบูลีนขนาด 1 บิต มีเกตอยู่สองประเภท คือ เกตอินพุตกับเกตคำนวณ

- เกตอินพุตไม่มีข้อมูลนำเข้าแต่จะแสดงถึงบิตของสายอักขระเลขฐานสองที่ส่งมาโดยอลิซหรือบ๊อบ
  - มีเกตอินพุตอยู่ $l_A + l_B$ เกต (กำกับด้วยหมายเลข 0 ถึง $(l_A + l_B - 1)$) โดยที่ l_A, l_B คือความยาวของสายอักขระจากอลิซและบ๊อบตามลำดับ
  - สำหรับ $0 \le i < l_A$ นั้น ข้อมูลส่งออกของเกตที่ i คือบิตที่ i ของสายอักขระจากอลิซ
  - สำหรับ $0 \le i < l_B$ นั้น ข้อมูลส่งออกของเกตที่ $(i + l_A)$ คือบิตที่ i ของสายอักขระจากบ๊อบ
- เกตคำนวณแสดงถึงการคำนวณโดยมีมีข้อมูลนำเข้าสองช่อง
  - หมายเลขของเกตคำนวณเริ่มจาก $(l_A + l_B)$
  - สำหรับเกตคำนวณแต่ละเกต คุณจะต้องระบุหมายเลขของเกตที่เกี่ยวข้องจำนวน 2 เกตเพื่อใช้เป็นข้อมูลนำเข้า และ ระบุประเภทของการคำนวณ $p (0 \le p \le 15)$
    - เพื่อป้องกันการอ้างถึงแบบวงวน หมายเลขของเกตที่เกี่ยวข้องจะต้องน้อยกว่าหมายเลขของเกตคำนวณนั้น ๆ
    - หากให้ข้อมูลส่งออกของเกตที่เกี่ยวข้องทั้งสองเกตคือ x_0 และ x_1 ตามลำดับแล้ว (โดยที่ $x_0, x_1 \in \{0, 1\}$) ข้อมูลส่งออกของเกตคำนวณคือ

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

ขอให้ดูตัวอย่างต่อไปนี้

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

รายละเอียดการเขียนโปรแกรม

ให้สังเกตว่า

- อาเรย์ทั้งหมดเริ่มต้นด้วย 0 ตัวอย่างเช่น ถ้าให้ `a` เป็นอาเรย์ความยาว `n` แล้ว ข้อมูลที่ใช้งานได้คือ `a[0]` to `a[n-1]` การเข้าถึงข้อมูลช่องอื่น ๆ อาจจะทำให้เกิดปัญหา out-of-bounds
- สายอักขระทั้งหมดนั้นมีอักขระ null (`\0`) ปิดท้ายเสมอ

คุณจะต้องเขียนฟังก์ชันต่อไปนี้

อลิซ

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

ประเภท	ตัวแปร	ความยาว	ความหมาย	ข้อจำกัด
ข้อมูลนำเข้า	n	1	n	$0 \le n \le 700$
	names	n	รายชื่อของสมาชิกแต่ละคน	ชื่อทั้งหมดไม่ซ้ำกันเลย และประกอบด้วยตัวอักษรพิมพ์เล็กภาษาอังกฤษเท่านั้น และมีความยาวไม่เกิน 4 ตัวอักษร
	numbers	n	ตัวเลขที่ชอบของแต่ละคน	ตัวเลขแต่ละตัวมีค่าเป็นไปได้ตั้งแต่ 0 ถึง 65535
ข้อมูลส่งออก	outputs_alice	l_A	สายอักขระเลขฐานสองที่ส่งให้เซอร์กิต	
	(ค่าที่คืนจากฟังก์ชัน)	1	l_A	**ต้องทำให้ l_A ไม่เกิน 10^5 และถ้าหาก n มีค่าเหมือนเดิมแล้ว l_A จะต้องคงที่**

บ๊อบ

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

ประเภท	ตัวแปร	ความยาว	ความหมาย	ข้อจำกัด
ข้อมูลนำเข้า	m	1	m	$0 \le m \le 1000$
	senders	m	ชื่อของผู้ส่งของจดหมายแต่ละฉบับ	ชื่อทั้งหมดจะปรากฏอยู่ในข้อมูลนำเข้าของอลิซ
	recipients	m	ชื่อของผู้รับของจดหมายแต่ละฉบับ	
ข้อมูลส่งออก	outputs_bob	l_B	สายอักขระเลขฐานสองที่ส่งให้เซอร์กิต	
	(ค่าที่คืนจากฟังก์ชัน)	1	l_B	**ต้องทำให้ l_B ไม่เกิน 10^5 และหาก m มีค่าเหมือนเดิมแล้ว l_B จะต้องคงที่**

เซอร์กิต

เพื่อทำให้การคำนวณของเซอร์กิตเป็นเหมือนวงจรทั่วไปจริง ๆ คุณจะไม่สามารถเข้าถึงสายอักขระของอลิซหรือบ๊อบได้โดยตรง คุณจะทราบเพียงแค่ความยาวของสายอักขระทั้งสองและต้องส่งออกโครงสร้างของวงจร

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

ประเภท	ตัวแปร	ความยาว	ความหมาย	ข้อจำกัด
ข้อมูลนำเข้า	la	1	l_A	
	lb	1	l_B	

ประเภท	ตัวแปร	ความยาว	ความหมาย	ข้อจำกัด
ข้อมูลส่งออก	operations	l	ประเภทของการทำงานของเกตแต่ละเกตในวงจร	จำนวนเต็มมีค่าตั้งแต่ 0 ถึง 15
	operands	l	ข้อมูลที่ใช้ของเกตแต่ละเกตในวงจร	หมายเลขจะต้องน้อยกว่าหมายเลขของเกตปัจจุบัน
	outputs_circuit	n	หมายเลขเกตของข้อมูลส่งออกของวงจร	`outputs_circuit[i][j]` ระบุถึงบิตที่ j (นับจากบิตที่สำคัญน้อยที่สุด (least significant bit)) ของผลลัพธ์ของสมาชิกหมายเลข i โดยลำดับของสมาชิกเรียงตามข้อมูลนำเข้าของอลิซ
	(ค่าที่คืนจากฟังก์ชัน)	1	l ซึ่งแสดงถึงจำนวนเกตทั้งหมด (รวมเกตอินพุต)	**ต้องทำให้** $l \leq 2 \times 10^7$

ถึงแม้ว่าคุณสามารถแก้ไขข้อมูลของเกตที่มีหมายเลขน้อยกว่า $l_A + l_B$ ในอาเรย์ `operations` และ `operands` ได้ แต่ตัวตรวจจะไม่สนใจการแก้ไขดังกล่าว

ตัวอย่าง

ให้พิจารณาการเรียกใช้ดังต่อไปนี้

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

การเรียกใช้นี้หมายถึงสถานการณ์ดังต่อไปนี้

- อลิซรู้ว่ามีสมาชิก 3 คน โดยสมาชิกที่มีชื่อว่า `alic` มีตัวเลขที่ชอบเป็น 10000 เป็นต้น ข้อมูลส่งออกของ `alice()` แบบหนึ่งที่เป็นไปได้คือ
  - ฟังก์ชัน `alice()` คืนค่าเป็น 2 ซึ่งหมายความว่า $l_A = 2$
  - ในฟังก์ชัน `alice()` มีการตั้งค่า `outputs_alice[0] = 1, outputs_alice[1] = 0` เพื่อระบุว่าสายอักขระเลขฐานสองที่เป็นผลลัพธ์คือ 10.
- บ๊อบรู้ว่ามีตัวอักษรอยู่ 5 ตัว โดยตัวแรกส่งจาก `alic` ไปยัง `circ` เป็นต้น ข้อมูลส่งออกของ `bob()` แบบหนึ่งที่เป็นไปได้คือ
  - ฟังก์ชัน `bob()` คืนค่าเป็น 3 ซึ่งหมายความว่า $l_B = 3$
  - ในฟังก์ชัน `bob()` มีการตั้งค่า `outputs_bob[0] = 1, outputs_bob[1] = 1, outputs_bob[2] = 0` เพื่อระบุว่าสายอักขระเลขฐานสองที่เป็นผลลัพธ์คือ 110.

จากข้อมูลส่งออกของ `alice()` และ `bob()` ทำให้มีการเรียกฟังก์ชัน

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

ข้อมูลส่งออกที่ถูกต้องของฟังก์ชันนี้จะต้องเป็นดังนี้

- ฟังก์ชัน `circuit()` คืนค่าเป็น 7 ซึ่งหมายความว่าเราเพิ่มเกตคำนวณไปสองเกต ได้แก่หมายเลข 5 และ 6
- ฟังก์ชัน `circuit()` มีการตั้งค่า `operations`, `operands`, และ `outputs_circuit` ดังต่อไปนี้
  - `operations = {-1, -1, -1, -1, -1, 8, 14}` โดยเราใช้ `-1` เพื่อระบุว่าเราไม่สนใจข้อมูลจากเกตอินพุต
  - `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}, ...}` อาเรย์นี้ค่อนข้างยาว คุณสามารถดูไฟล์ `abc.cpp` ในไฟล์แนบสำหรับอาเรย์ทั้งหมด

จากข้อมูลดังกล่าว การคำนวณจะเป็นดังนี้

- เพิ่มเกตคำนวณประเภท 8 โดยใช้ข้อมูลนำเข้าจากเกต 0 และ 4 โดยข้อมูลส่งออกของเกต 0 คือบิตที่ 0 ของสายอักขระจากอลิซ ซึ่งคือค่า 1 ส่วนข้อมูลส่งออกของเกต 4 คือ บิตที่ 2 ของสายอักขระจากบ๊อบซึ่งคือ 0 ทำให้ข้อมูลส่งออกของเกต 5 คือ $f(8, 0, 1) = 0 \text{ AND } 1 = 0$.
- เพิ่มเกตคำนวณประเภท 14 โดยใช้ข้อมูลนำเข้าจากเกต 2 และ 5 โดยข้อมูลส่งออกของเกต 2 คือบิตที่ 0 ของสายอักขระจากบ๊อบ ซึ่งคือค่า 1 ส่วนข้อมูลส่งออกของเกต 5 คือ 0 ทำให้ข้อมูลส่งออกของเกต 6 คือ $f(14, 1, 0) = 1 \text{ OR } 0 = 1$.
- `output_circuit[0]` อธิบายถึงผลลัพธ์สุดท้ายของ `alic` ซึ่งคือ $(0100111000100000)_2 = 20000$ เนื่องจาก `alic` ได้รับจดหมายจาก `bob` เท่านั้น ทำให้ผลลัพธ์ของ `alic` คือ 20000.
- ผลลัพธ์ของ `bob` ควรจะเป็น 0 เนื่องจากบ๊อบไม่ได้รับจดหมายเลย ผลลัพธ์ของ `circ` ควรเป็น $(10000 + 20000 + 30000 + 30000) \bmod 65536 = 24464$

ไฟล์ `abc.cpp` ในเอกสารแนบจะสามารถทำงานตัวอย่างนี้ได้อย่างถูกต้อง แต่ไม่รับประกันว่าจะทำงานผ่านข้อมูลทดสอบอื่น ๆ

เงื่อนไข

สำหรับทุก ๆ ข้อมูลทดสอบนั้น

- $0 \le n \le 700, 0 \le m \le 1000$.
- ชื่อคนทั้งหมดไม่ซ้ำกันเลย และประกอบด้วยตัวอักษรพิมพ์เล็กภาษาอังกฤษเท่านั้น และมีความยาวไม่เกิน 4 ตัวอักษร
- ตัวเลขที่ชอบของสมาชิกแต่ละคนอยู่ในช่วง 0 ถึง 65535
- ชื่อของผู้ส่งและผู้รับทั้งหมดจะอยู่ในอาเรย์นำเข้าชื่อ `names` ของอลิซ
- `alice()` และ `bob()` ใช้หน่วยความจำได้ไม่เกินเป็น 2048 MiB และต้องทำงานเสร็จภายในเวลา 0.02 วินาที
- `circuit()` ใช้หน่วยความจำได้ไม่เกิน 2048 MiB และต้องทำงานเสร็จภายใน 7 seconds.

ในการตรวจนั้น `alice()` และ `bob()` อาจจะถูกเรียกใช้ได้หลายครั้งในข้อมูลทดสอบแต่ละอัน ข้อจำกัดของเวลา 0.02 วินาทีนั้นเป็นต่อหนึ่งการเรียกฟังก์ชันแต่ละครั้ง

ชุดข้อมูลทดสอบ

ชุดข้อมูลทดสอบประเภท A (12 แต้ม)

ชุดข้อมูลทดสอบ 1,2,3 เป็นชุดข้อมูลทดสอบประเภท A ซึ่งรับประกันว่า $n = 1$

แต่ละชุดข้อมูลทดสอบมีข้อจำกัดเพิ่มเติมดังนี้

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

ชุดข้อมูลทดสอบ Type B (54 คะแนน)

ชุดข้อมูลทดสอบ 4,5,6 เป็นชุดข้อมูลทดสอบประเภท B ซึ่งมีเป็นดังนี้

- $0 \le n \le 30, \frac{n}{2} \le m \le n^2$
- ไม่มีจดหมายสองฉบับใดที่มีรายชื่อผู้ส่งและผู้รับซ้ำเหมือนกัน
- ชื่อสมาชิกทั้งหมดปรากฏอยู่ในข้อมูลนำเข้าของบ๊อบ (กล่าวคือ สมาชิกทุกคนส่งหรือรับจดหมายอย่างน้อยหนึ่งฉบับ)

แต่ละชุดข้อมูลทดสอบมีข้อจำกัดเพิ่มเติมดังนี้

- ชุดข้อมูลทดสอบ 4 (24 คะแนน): $n = 26$ ชื่อสมาชิกทุกคนเป็นตัวอักษรพิมพ์เล็กตัวเดียว และปรากฏเรียงตามลำดับจาก `a` ถึง `z` ในข้อมูลนำเข้าของอลิซ
- ชุดข้อมูลทดสอบ 5 (24 คะแนน): $n = 26$
- ชุดข้อมูลทดสอบ 6 (6 คะแนน): ไม่มีข้อจำกัดเพิ่มเติม

ชุดข้อมูลทดสอบ Type C (34 คะแนน)

ชุดข้อมูลทดสอบ 7,8,9 เป็นชุดข้อมูลทดสอบประเภท C ซึ่งรับประกันว่า $0 \le n \le 700, 0 \le m \le 1000$.

แต่ละชุดข้อมูลทดสอบมีข้อจำกัดเพิ่มเติมดังนี้

- ชุดข้อมูลทดสอบ 7 (18 คะแนน): $n = 676$ ชื่อสมาชิกทุกคนเป็นตัวอักษรพิมพ์เล็กสองตัว และปรากฏเรียงตามลำดับตามแบบตัวอักษร (ซึ่งคือ `aa`, `ab`, `ac`, ..., `az`, `ba`, ..., `bz`, `ca`, ..., `zz`)
- ชุดข้อมูลทดสอบ 8 (10 คะแนน): $n = 676$.
- ชุดข้อมูลทดสอบ 9 (6 คะแนน): ไม่มี่ข้อจำกัดเพิ่มเติม

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างอ่านข้อมูลนำเข้าในรูปแบบต่อไปนี้

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

เกรดเดอร์ตัวอย่างส่งข้อมูลส่งออกดังนี้

- ถ้าโปรแกรมจบการทำงานอย่างถูกต้อง เกรดเดอร์ตัวอย่างจะพิมพ์ข้อมูล n บรรทัด แต่ละบรรทัดประกอบด้วยจำนวนเต็มหนึ่งตัวซึ่งระบุถึงผลลัพธ์ที่คำนวณโดยฟังก์ชันของคุณของสมาชิกแต่ละคน
- ถ้าไม่เช่นนั้นเกรดเดอร์ตัวอย่างจะไม่แสดงข้อมูลใด ๆ ไปยัง stdout และพิมพ์ข้อความแสดงความผิดพลาดลงไปในไฟล์ `abc.log` ในไดเร็กทอรี่นั้น
- นอกจากนี้ เกรดเดอร์ตัวอย่างจะแสดงค่าของ l_A, l_B, l และเวลาในการทำงานของแต่ละฟังก์ชันลงในไฟล์ `abc.log` ด้วย

เกรดเดอร์ตัวอย่างจะไม่ตรวจสอบหน่วยความจำ และข้อจำกัดที่ระบุว่าสำหรับ n / m ที่เหมือนเดิม l_A / l_B จะต้องคงที่