

S_0	S_1	S_2	S_3	S_4	S_5	สถานะประตู (เรียงจาก 0, 1, 2)	ประตูบานหน้าต่างสุดท้ายที่เห็น
on	on	on	on	on	on	ปิด,ปิด,เปิด	0
on	on	on	off	on	off	เปิด,ปิด,ปิด	1
on	on	on	off	on	on	เปิด,ปิด,เปิด	1
off	on	off	off	on	off	เปิด,เปิด,ปิด	2
off	on	off	off	on	on	เปิด,เปิด,เปิด	ไม่มี (เปิดทุกบานแล้ว)

รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันต่อไปนี้

```
void solve(int N)
```

- พารามิเตอร์ N แทนจำนวนประตู
- ฟังก์ชันนี้จะถูกเรียกหนึ่งครั้ง
- ฟังก์ชันนี้จะต้องพยายามหาทางเปิดประตูทั้ง N บาน

ฟังก์ชัน solve จะสามารถเรียกฟังก์ชัน control_switches ได้ไม่เกิน Q ครั้ง

```
int control_switches(vector<int> V)
```

- ฟังก์ชันนี้จะถูกเรียกได้ไม่เกิน Q ครั้ง ถ้าเรียกเกินโปรแกรมจะจบการทำงานและถือว่าเกิดความผิดพลาด
- อาร์เรย์ V เป็นอาร์เรย์ขนาด $2N$ ที่แต่ละช่องมีค่า 1 หรือ 0 ที่แทนสถานะของสวิตช์ เราจะกำหนดให้ $S_i = \text{on}$ เมื่อ $V[i] = 1$ และ $S_i = \text{off}$ เมื่อ $V[i] = 0$
- ฟังก์ชันจะคืนหมายเลขของประตูหน้าต่างสุดท้ายที่ปิดอยู่ ถ้าประตูเปิดหมดแล้วโปรแกรมจะทำงานสำเร็จและจบการทำงาน

เงื่อนไข

- $1 \leq N \leq 500$
- $Q = 15\,000$

ปัญหาย่อย

1. (10 points) $N \leq 10$
2. (5 points) สำหรับทุกๆ ประตู i จะมีจำนวนเต็ม j ที่สวิตช์ S_{2j} และ S_{2j+1} ควบคุมประตุนั้นอยู่, และ $N \leq 80$
3. (7 points) สำหรับทุกๆ ประตู i จะมีจำนวนเต็ม j ที่สวิตช์ S_{2j} และ S_{2j+1} ควบคุมประตุนั้นอยู่, และ $N \leq$

160

4. (8 points) สำหรับทุกๆ ประตู่ i จะมีจำนวนเต็ม j ที่สวิตช์ S_{2j} และ S_{2j+1} ควบคุมประตู่ นั้นอยู่
5. (10 points) $N \leq 200$
6. (19 points) $N \leq 300$
7. (20 points) $N \leq 400$
8. (21 points) ไม่มีเงื่อนไขเพิ่มเติมอื่น ๆ

ตัวอย่าง

สำหรับตัวอย่างข้างต้น เกรดเดอร์จะเรียก

```
solve(3)
```

ฟังก์ชัน solve จะสามารถเรียกฟังก์ชัน control_switches ได้ โดยค่าที่ตอบมาสำหรับการเรียกแต่ละครั้ง จะเป็นดังต่อไปนี้

```
control_switches([1, 1, 1, 1, 1, 1])
```

สถานะของสวิตช์คือทุกสวิตช์นั้น on ซึ่งจะทำให้ประตู่ 0 และ 1 ปิด ส่วนประตู่ 2 นั้นเปิดเนื่องจาก $S_4 = S_5 = \text{on}$ เนื่องจากประตู่หน้าสุดที่ปิดอยู่คือประตู่ 0 ดังนั้นฟังก์ชัน control_switch จะคืนค่า 0

```
control_switches([1, 1, 1, 0, 1, 0])
```

เนื่องจาก $S_1 = \text{on}$ และ $S_3 = \text{off}$ ประตู่ 0 จะเปิด ส่วนประตู่ 1 และ 2 นั้นปิด เนื่องจากประตู่หน้าสุดที่ปิดอยู่คือประตู่ 1 ดังนั้นฟังก์ชัน control_switch จะคืนค่า 1

```
control_switches([1, 1, 1, 0, 1, 1])
```

เนื่องจาก $S_1 = \text{on}$ และ $S_3 = \text{off}$ ประตู่ 0 จะเปิด นอกจากนี้ เนื่องจาก $S_4 = S_5 = \text{on}$ ทำให้ประตู่ 2 ก็เปิด ส่วนประตู่ 1 ยังปิดอยู่ เนื่องจากประตู่หน้าสุดที่ปิดอยู่คือประตู่ 1 ดังนั้นฟังก์ชัน control_switch จะคืนค่า 1

```
control_switches([0, 1, 0, 0, 1, 0])
```

เนื่องจาก $S_1 = \text{on}$ และ $S_3 = \text{off}$ ประตู่ 0 จะเปิด นอกจากนี้ เนื่องจาก $S_0 = S_2 = \text{off}$ ทำให้ประตู่ 1 ก็เปิด ส่วนประตู่ 2 ยังปิดอยู่ เนื่องจากประตู่หน้าสุดที่ปิดอยู่คือประตู่ 2 ดังนั้นฟังก์ชัน control_switch จะคืนค่า 2

```
control_switches([0, 1, 0, 0, 1, 1])
```

เนื่องจากสถานะของสวิตช์ที่ส่งมานี้ ทำให้ทุกประตูเปิดขึ้น ฟังก์ชัน `control_switches` จะถือว่าคุณทำงานสำเร็จและจบการทำงาน

เกรตเตอร์ตัวอย่าง

เกรตเตอร์ตัวอย่างจะอ่านข้อมูลดังนี้:

- บรรทัด 1: N
- บรรทัด $2 + i$ (สำหรับ $0 \leq i < N$): $a_i \ b_i \ x_i \ y_i$
เป็นข้อมูลของประตู i โดยที่ a_i และ b_i แทนหมายเลขสวิตช์ ($0 \leq a_i < 2N, 0 \leq b_i < 2N$) และประตู i จะเปิดก็ต่อเมื่อ $S_{a_i} = x_i$ และ $S_{b_i} = y_i$ โดยที่ x_i กับ y_i จะเป็น 1 เพื่อแทนสถานะ on และเป็น 0 เพื่อแทนสถานะ off

ถ้าเรียกฟังก์ชัน `control_switches` เกิน Q ครั้ง เกรตเตอร์จะแสดงความผิดพลาด ถ้าเรียกไม่เกินและสามารถเปิดประตูได้ เกรตเตอร์ตัวอย่างจะแสดงจำนวนครั้งที่เรียก `control_switches`

ขอบเขต

- Time limit: 1 seconds
- Memory limit: 512 MB