# การแล่นเรือใบ (sailing)

การแล่นเรือใบ (หรือ sailing) คือการใช้แรงลมในการขับเคลื่อนเรือบนผิวน้ำ แต่วันนี้เป็นวันสอบคอมพิวเตอร์ดังนั้นเรื่องเรือใบเอาไว้ก่อน

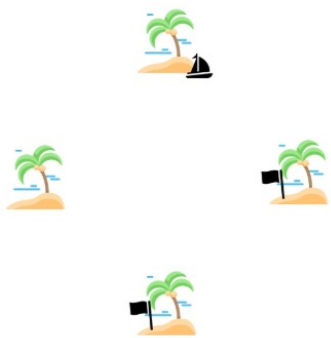
คุณเป็นนักเดินเรือคนหนึ่งที่ออกจากฝั่งมาเพื่อหาสมบัติที่เป็นความลับของโลกใบนี้ แต่ทว่าคุณดันหลุดเข้ามาในหมู่เกาะที่น่าฉงน สิ่งที่คุณรู้คือหมู่เกาะนี้ประกอบไปด้วยเกาะที่หน้าตาเหมือนกัน $N$ เกาะเรียกว่าเป็นเกาะที่ $0$ ถึงเกาะที่ $N-1$ เรียงกันในลักษณะวงกลมโดยปัจจุบันคุณอยู่ที่เกาะที่ $0$ นอกจากนี้แต่ละเกาะยังสามารถมีหรือไม่มีธงอยู่บนเกาะก็ได้ นั่นคือแต่ละเกาะจะมีธงอยู่ $0$ หรือ $1$ ผืนเท่านั้น และ**เกาะที่ $0$ ไม่มีธงปักอยู่**

โชคยังดีที่คุณได้เอาธงที่เหมือนกับธงบนเกาะในตอนแรกเริ่ม (หมายความว่าธงทั้งหมดมีลักษณะเหมือนกัน ไม่สามารถแยกออกได้) มาด้วยเป็นจำนวนมากจนเรียกว่าไม่จำกัดเลยก็ได้ ดังนั้นเพื่อเป็นการทำเครื่องหมายไว้ คุณสามารถ *ปักธง**หรือ**เก็บธง* จากเกาะที่คุณอยู่ได้ ทั้งนี้ทุกเกาะสามารถมีธงได้**เพียงผืนเดียวเท่านั้น**

คุณสามารถเดินเรือไปยังเกาะถัดไปหรือเดินเรือไปยังเกาะก่อนหน้าได้ เนื่องจากลักษณะของหมู่เกาะที่เป็นวงกลม หากเดินเรือไปยังเกาะถัดไปที่เกาะ $N-1$ จะทำให้คุณไปโผล่ที่เกาะ $0$ แทน นอกนั้นจะเป็นการเดินเรือจากเกาะที่ $i$ ไปยังเกาะที่ $i+1$ ทำนองเดียวกันหากเดินเรือไปยังเกาะก่อนหน้าที่เกาะ $0$ ก็จะทำให้คุณไปโผล่ที่เกาะ $N-1$ นอกนั้นจะเป็นการเดินเรือจากเกาะที่ $i$ ไปยังเกาะที่ $i-1$

เพื่อหนีออกจากหมู่เกาะที่น่าฉงนนี้คุณต้องการรู้จำนวนเกาะว่ามีทั้งหมดกี่เกาะกันแน่ จากนั้นคุณจะได้ทำการคำนวณทิศทางการเดินเรือได้อย่างถูกต้อง อย่างไรก็ตามในท้องทะเลนั้นทรัพยาการบนเรือมีจำกัด คุณสามารถเดินเรือทั้งสองทิศทางได้ไม่เกิน $Q$ ครั้งก่อนที่จะอดตายอยู่บนเรือ

ตัวอย่างข้างต้นแสดงรูปแบบของหมู่เกาะที่มี $N=4$ เกาะและมี $f=[0,1,1,0]$ เมื่อ $f$ เป็น array ขนาด $N$ ที่ช่องที่ $i$ แทนว่าเกาะที่ $i$ มีธงอยู่หรือไม่ (0 แทนไม่มีธงและ 1 แทนมีธง) ซึ่งทั้งสองอย่างจะเป็นสิ่งคุณจะไม่ทราบในตอนที่โปรแกรมทำงาน



สามารถดูตัวอย่างการทำงานข้อมูลนี้ได้ที่หัวข้อตัวอย่างในหน้าถัดๆไป

สำหรับแต่ละข้อมูลชุดทดสอบ คุณจะต้องเขียนโปรแกรมเพื่อหาจำนวนเกาะทั้งหมด สำหรับการทำงานหนึ่งครั้งอาจจะมีการแก้ปัญหาชุดทดสอบหลายอัน ให้พิจารณาว่าเป็นปัญหาที่เป็นอิสระต่อกัน

# รายละเอียดการเขียนโปรแกรม

**หมายเหตุ**: ในการทำงานแต่ละครั้งของโปรแกรม เกรดเดอร์อาจจะมีการเรียกฟังก์ชันหลัก (`set_sail`) เพื่อทำงานกับชุดทดสอบหลายชุด ในการเรียกแต่ละครั้งให้พิจารณาว่าเป็นการทำงานที่**เป็นอิสระต่อกัน** ดังนั้นผู้เข้าแข่งขันจะต้องจัดการกับค่าเก่าที่ค้างจากการเรียกในครั้งก่อนเอง เช่น พวกค่าในตัวแปร global เป็นต้น

คุณจะต้องเขียนฟังก์ชันดังต่อไปนี้

```
int set_sail()
```

- ฟังก์ชันนี้จะถูกเรียกหนึ่งครั้งสำหรับแต่ละข้อมูลชุดทดสอบ แต่เป็นไปได้ที่ในการทำงานหนึ่งครั้ง จะมีการเรียกฟังก์ชันให้ทำงานกับหลายชุดทดสอบก็ได้ ให้พิจารณาว่าเป็นปัญหาที่เป็นอิสระต่อกัน
- สถานะของธงที่ปักอยู่ตามเกาะต่าง ๆ จะถูกกำหนดไว้ก่อนเริ่มเรียกฟังก์ชันนี้ (สถานะของธงไม่ได้เป็นการกำหนดค่าแบบเปลี่ยนแปลงได้)
- เกรดเดอร์จะเรียกฟังก์ชันนี้เพื่อแทนการเริ่มออกเดินเรือจากเกาะที่ $0$
- ฟังก์ชันจะต้องคืนค่าจำนวนเกาะที่อยู่ในหมู่เกาะแห่งนี้

ฟังก์ชัน `set_sail` สามารถเรียกใช้งาน $3$ ฟังก์ชันต่อไปนี้ได้

```
boolean sail_forward()
```

- แทนการเดินเรือไปข้างหน้า โดยจะคืนค่า false หากเกาะที่เดินเรือไปไม่มีธง และ true หากเกาะที่เดินทางไปมีธงอยู่
- โปรแกรมดังกล่าวจะต้องเรียกฟังกันนี้ รวมกับฟังก์ชัน sail_backward ไม่เกิน $Q$ ครั้งจึงจะได้คะแนน

```
boolean sail_backward()
```

- แทนการเดินเรือไปข้างหลัง โดยจะคืนค่า false หากเกาะที่เดินเรือไปไม่มีธง และ true หากเกาะที่เดินทางไปมีธงอยู่
- โปรแกรมดังกล่าวจะต้องเรียกฟังกันนี้ รวมกับฟังก์ชัน sail_forward ไม่เกิน $Q$ ครั้งจึงจะได้คะแนน

```
void flag()
```

- แทนการปักหรือเก็บธงที่เกาะที่อยู่ โดยหากเกาะปัจจุบันมีธงอยู่แล้วจะทำการเก็บธงที่เกาะนั้น ๆ มิเช่นนั้นจะทำการปักธงที่เกาะนั้น ๆ

# เงื่อนไข

- สำหรับแต่ละข้อมูลทดสอบ $2 \le N \le 1\,000\,000$
- ตลอดการทำงานหนึ่งครั้ง $\sum N \le 10\,000\,000$

# ปัญหาย่อย

1. (12 คะแนน) $Q = N(N+1), \sum N \le 5\,000$
2. (33 คะแนน) $N > 32, Q = 10N + 160$
3. (55 คะแนน) $Q = 10N + 160$

ในปัญหาย่อยที่ $2$ และ $3$ กำหนดให้ $S$ แทนจำนวนครั้งที่เรียกใช้ฟังก์ชัน `sail_forward` และ `sail_backward` และให้ $T = \dfrac{S - 160}{N}$ จะให้คะแนนดังต่อไปนี้

| เงื่อนไข | อัตราส่วนการให้คะแนน |
|---|---|
| $10 < T$ | $0$ |
| $8 < T \le 10$ | $-0.217T + 2.171$ |
| $1 < T \le 8$ | $1 - 0.188 \log_2 T$ |
| $T \le 1$ | $1$ |

แล้วคะแนนในชุดทดสอบนั้นจะมีค่าเท่ากับ **อัตราส่วนคะแนน** (จากตาราง) คูณด้วยคะแนนเต็มของปัญหาย่อยนั้น และคะแนนในแต่ละปัญหาย่อยจะเป็น **ค่าน้อยสุด** ของคะแนนในแต่ละครั้งที่มีการเรียก `set_sail()` ในชุดทดสอบของปัญหาย่อยนั้น

# ตัวอย่าง

พิจารณาตัวอย่างมีเกาะอยู่ $N = 4$ เกาะ และมีสถานะการปักธงแทนด้วย $[0, 1, 1, 0]$ เมื่อ $1$ แทนว่าเกาะนั้นมีธง และ $0$ แทนว่าเกาะนั้นไม่มีธง

ตัวอย่างการทำงานของโปรแกรมจะเป็นดังนี้ เกรดเดอร์จะเรียก

```
set_sail()
```

เพื่อออกเดินเรือ โดยปัจจุบันอยู่ที่เกาะ $0$ จากนั้นฟังก์ชัน `set_sail` อาจจะเรียก

```
flag()
```

เพื่อทำการปักธงที่เกาะ $0$ ปัจจุบันลักษณะธงในแต่ละเกาะเป็น $[1, 1, 1, 0]$

จากนั้น ถ้าเรียก

```
sail_forward()
```

จะมีการแล่นเรือไปยังเกาะ $1$ ฟังก์ชันจะคืนค่า `true` เพราะว่าเกาะที่ $1$ มีธงอยู่ จากนั้นถ้าเรียก

```
flag()
```

จะเก็บธงที่เกาะ $1$ ทำให้ปัจจุบันมีลักษณะธงในแต่ละเกาะเป็น $[1, 0, 1, 0]$ เมื่อเรียก

```
sail_forward()
```

จะแล่นเรือไปยังเกาะ $2$ ฟังก์ชันจะคืนค่า `true` เพราะว่าเกาะที่ $2$ มีธงอยู่ จากนั้นถ้าเราเรียก

```
flag()
```

เราจะเก็บธงที่เกาะ $2$ ปัจจุบันมีลักษณะธงในแต่ละเกาะจะเป็น $[1, 0, 0, 0]$ หลังจากนั้นเราสามารถเรียก

```
sail_backward()
```

เพื่อแล่นเรือกลับมายังเกาะ $1$ ฟังก์ชันจะคืนค่า `false` เพราะว่าเกาะที่ $1$ ไม่มีธงอยู่ หลังจากนั้นถ้าเรียก

```
sail_backward()
```

เราจะแล่นเรือกลับมายังเกาะ $0$ ฟังก์ชันจะคืนค่า `true` เพราะเกาะที่ $0$ มีธงอยู่

ฟังก์ชัน `set_sail` ที่ถูกต้องจะต้องคืนค่า $4$ แทนว่าในหมู่เกาะนี้มีเกาะอยู่ $4$ เกาะ

## เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างจะอ่านข้อมูลนำเข้าดังนี้:

- บรรทัด 1: $T$ เมื่อ $T$ เป็นจำนวนชุดทดสอบ

จากนั้นสำหรับแต่ละชุดทดสอบ จะประกอบด้วยข้อมูลสองบรรทัด ในรูปแบบดังนี้

- บรรทัด 1: $N \;\; Q$
- บรรทัด 2: $f_0 \; f_1 \; f_2 \; \ldots \; f_{N-1}$
  โดยที่ $f_i \in \{0, 1\}$ แทนสถานะเริ่มต้นของธง รับประกันว่า $f_0 = 0$

ถ้าถามเกิน $Q$ ครั้ง เกรดเดอร์ตัวอย่างจะพิมพ์ข้อความแสดงความผิดพลาดถ้าไม่เช่นนั้นเกรดเดอร์จะตรวจค่าที่ คืนมาจากฟังก์ชัน `set_sail` และพิมพ์ว่าคำตอบถูกหรือไม่ พร้อมจำนวนครั้งที่เรียก `sail_forward` และ `sail_backward`

## ขอบเขต

- Time limit: 2.5 seconds
- Memory limit: 128 MB