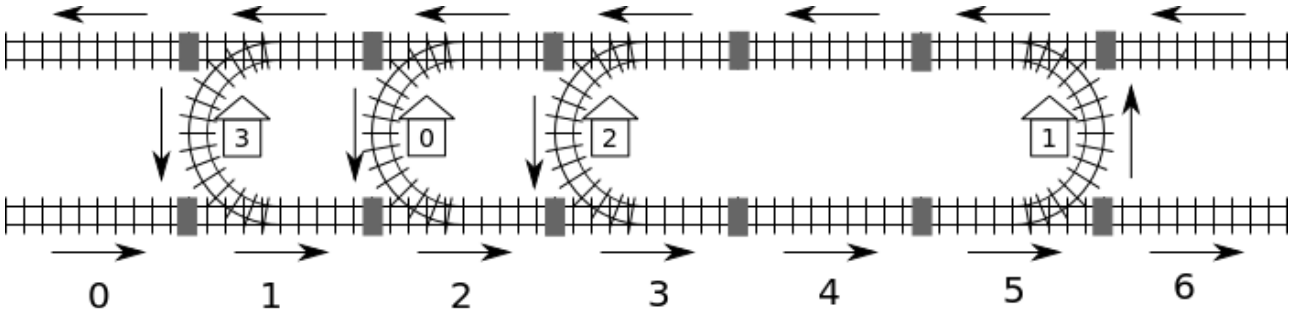
Rail

ไต้หวันมีระบบเส้นทางรถไฟขนาดใหญ่เชื่อมต่อระหว่างชายฝั่งตะวันตกและชายฝั่งตะวันออกของเกาะ เส้นทางรถไฟดังกล่าวแบ่งเป็นบล็อกจำนวน m บล็อก บล็อกเหล่านี้อยู่ต่อเนื่องกันและมีหมายเลขเป็น $0, \ldots, m-1$ โดยเริ่มนับจากปลายด้านทิศตะวันตก แต่ละบล็อกประกอบด้วยรางรถไฟด้านเหนือและรางรถไฟด้านใต้ รางรถไฟด้านเหนือมีไว้สำหรับใช้วิ่งทางเดียวไปทางทิศตะวันตก รางรถไฟด้านใต้สำหรับใช้วิ่งทางเดียวไปทางทิศตะวันออก ระหว่างรางสองรางนี้อาจจะมีสถานีรถไฟตั้งอยู่ก็ได้

บล็อกมีสามประเภทดังนี้ บล็อกประเภท *C* จะมีสถานีรถไฟตั้งอยู่ ในการจะเข้าไปยังสถานีดังกล่าว รถไฟจะต้องเดินทางจากรางรถไฟด้านเหนือและวนออกไปยังรางรถไฟด้านใต้ สำหรับบล็อกประเภท *D* จะมีสถานีรถไฟตั้งอยู่เช่นเดียวกัน แต่ในการจะเข้าไปยังสถานีดังกล่าว รถไฟจะต้องเดินทางจากรางรถไฟด้านใต้และวนออกไปยังรางรถไฟด้านเหนือ สุดท้ายบล็อกประเภท *empty* จะไม่มีสถานีรถไฟอยู่ ให้พิจารณาตัวอย่างด้านล่างที่บล็อกหมายเลข 0, 4, และ 6 เป็นบล็อกประเภท empty, บล็อกหมายเลข 1, 2, และ 3 เป็นบล็อกประเภท C และบล็อกหมายเลข 5 เป็นบล็อกประเภท D แต่ละบล็อกจะเชื่อมต่อกันในแนวนอน รางรถไฟที่อยู่บนบล็อกที่ติดกันจะเชื่อมกันด้วย *จุดเชื่อมต่อ* ซึ่งจะแสดงเป็นสีเหลี่ยมสีเทาในรูปถัดไป



ระบบรถไฟมีสถานีรถไฟทั้งสิ้น n สถานี มีหมายเลขจาก 0 ถึง $n-1$ เรารับประกันว่า *เราสามารถเดินทางจากสถานีใด ๆ ไปยังอีกสถานีใด ๆ ก็ได้* ผ่านทางรางรถไฟเหล่านี้ ยกตัวอย่างเช่น เราสามารถเดินทางจากสถานี 0 ไปยังสถานี 2 โดยเริ่มที่บล็อกหมายเลข 2 จากนั้นเดินทางผ่านบล็อกหมายเลข 3 และ 4 ทางรางรถไฟด้านใต้ และเดินทางผ่านบล็อกหมายเลข 5 โดยผ่านทางสถานี 1 จากนั้นเดินทางผ่านบล็อกหมายเลข 4 ทางรางรถไฟด้านเหนือและไปถึงสถานี 2 ที่บล็อกหมายเลข 3

เนื่องจากการเดินทางเป็นไปได้หลายแบบ เราจะนิยามระยะทางระหว่างสถานีหนึ่งไปยังอีกสถานีหนึ่งว่าเป็นจำนวนจุดเชื่อมต่อที่ *น้อยที่สุด*ที่รถไฟจะต้องวิ่งผ่านตามเส้นทางนั้น ตัวอย่างเช่น เส้นทางที่สั้นที่สุดจากสถานี 0 ไปยังสถานี 2 คือการเดินทางผ่านบล็อก 2-3-4-5-4-3 เส้นทางดังกล่าวผ่านจุดเชื่อมต่อ 5 จุด ดังนั้นจะมีระยะทางเท่ากับ 5

เรามีระบบคอมพิวเตอร์ที่จัดการระบบรถไฟนี้ แต่โชคไม่ดีที่เกิดปัญหาระบบไฟฟ้าทำให้คอมพิวเตอร์ไม่ทราบว่าสถานีต่าง ๆ อยู่ที่ใดและประเภทของบล็อกที่สถานีต่าง ๆ อยู่นั้นเป็นประเภทใด สิ่งเดียวที่คอมพิวเตอร์ทราบคือสถานี 0 อยู่ที่บล็อกหมายเลขใด และบล็อกดังกล่าวเป็นบล็อกประเภท C ยังดีที่คอมพิวเตอร์สามารถสอบถามระยะทางระหว่างสถานีหนึ่ง ๆ ไปยังอีกสถานีใด ๆ ได้ ตัวอย่างเช่น คอมพิวเตอร์สามารถสอบถามว่า 'ระยะทางระหว่างสถานี 0 และสถานี 2 เป็นเท่าใด?' คำตอบที่จะได้รับคือ 5

งานของคุณ

คุณต้องเขียนฟังก์ชัน `findLocation` เมื่อคำนวณหมายเลขบล็อกและประเภทของบล็อกสำหรับทุก ๆ สถานี

- `findLocation(n, first, location, stype)`
  - `n`: จำนวนของสถานี
  - `first`: หมายเลขบล็อกของสถานี 0
  - `location`: อาร์เรย์ขนาด n; คุณต้องใส่ค่าหมายเลขบล็อกของสถานี i ใน `location[i]`.
  - `stype`: อาร์เรย์ขนาด n; คุณจะต้องระบุประเภทบล็อกของสถานี i ใน `stype[i]` โดยที่ให้มีค่า 1 ถ้าเป็นประเภท C และ 2 ถ้าเป็นประเภท D

คุณสามารถเรียกใช้ฟังก์ชัน `getDistance` เพื่อที่จะช่วยในการคำนวณตำแหน่งและประเภทของบล็อกของสถานีต่าง ๆ

- `getDistance(i, j)` คืนระยะทางจากสถานี `i` ไปยังสถานี `j` นอกจากนี้ `getDistance(i, i)` จะคืนค่า 0 และ `getDistance(i, j)` จะคืนค่า -1 ถ้า `i` หรือ `j` อยู่นอกขอบเขต $0 \le i, j \le n-1$

ปัญหาย่อย

ในทุก ๆ ปัญหาย่อย จำนวนของบล็อก m จะมีไม่เกิน 1,000,000 เสมอ ในบางปัญหาย่อยจำนวนครั้งในการเรียก `getDistance` จะมีจำกัด ซึ่งขอบเขตนี้เปลี่ยนแปลงตามปัญหาย่อย โปรแกรมของคุณจะได้รับผลการตรวจว่าคำตอบผิด ('wrong answer') ถ้าคุณเรียกใช้เกินขอบเขต

ปัญหาย่อย	คะแนน	n	จำนวนครั้งการเรียก getDistance	หมายเหตุ
1	8	$1 \le n \le 100$	ไม่จำกัด	ทุก ๆ สถานียกเว้นสถานี 0 จะอยู่ในบล็อกประเภท D
2	22	$1 \le n \le 100$	ไม่จำกัด	ทุก ๆ สถานีทางด้านตะวันออกของสถานี 0 เป็นประเภท D ทุกสถานีทางด้านตะวันตกของสถานี 0 เป็นประเภท C
3	26	$1 \le n \le 5,000$	$n(n-1)/2$	ไม่มีขีดจำกัดเพิ่มเติม
4	44	$1 \le n \le 5,000$	$3(n-1)$	ไม่มีขีดจำกัดเพิ่มเติม

รายละเอียดการเขียนโปรแกรม

คุณจะต้องส่งไฟล์หนึ่งไฟล์เท่านั้น โดยอาจจะเป็นไฟล์ชื่อ `rail.c`, `rail.cpp` หรือ `rail.pas` โดยที่ไฟล์นี้จะต้องมีฟังก์ชัน `findLocation` ที่ทำงานได้ตามที่ระบุไว้ข้างต้น คุณจะต้อง include header `rail.h` ในไฟล์ดังกล่าวด้วยถ้าคุณใช้ภาษา C/C++

โปรแกรมภาษา C/C++

```
void findLocation(int n, int first, int location[], int stype[]);
```

โปรแกรมภาษาปาสคาล

```
procedure findLocation(n, first : longint; var location,
stype : array of longint);
```

รูปแบบของฟังก์ชัน `getDistance` เป็นดังนี้

โปรแกรมภาษา C/C++

```
int getDistance(int i, int j);
```

โปรแกรมภาษาปาสคาล

```
function getDistance(i, j: longint): longint;
```

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างจะอ่านข้อมูลนำเข้าในรูปแบบต่อไปนี้:

- บรรทัด 1: หมายเลขปัญหาย่อย
- บรรทัด 2: `n`
- บรรทัด $3+i, (0 <= i <= n-1)$: `stype[i]` (1 สำหรับประเภท C และ 2 สำหรับประเภท D), `location[i]`

เกรดเดอร์ตัวอย่างจะพิมพ์ `Correct` ถ้า `location[0]` ... `location[n-1]` และ `stype[0]` ... `stype[n-1]` ที่คำนวณจากโปรแกรมของคุณตรงกับข้อมูลนำเข้าเมื่อฟังก์ชัน `findLocation` ของคุณทำงานเสร็จสิ้น หรือจะพิมพ์ว่า `Incorrect` ถ้าค่าเหล่านี้ไม่ตรงกัน