ตาชั่ง

อามินามีเหรียญ 6 เหรียญ คิดเป็นเหรียญหมายเลข 1 ถึง 6 เธอทราบว่าเหรียญทุกเหรียญมีน้ำหนักแตกต่างกัน เธอต้องการเรียงเหรียญทั้งหมดตามน้ำหนัก เพื่อทำให้บรรลุเป้าหมายนี้ เธอจึงคิดค้นตาชั่งแบบใหม่

ตาชั่งทั่วไปจะมีจานสองจาน ในการใช้งาน คุณจะวางเหรียญลงในแต่ละจาน จากนั้นตาชั่งจะบอกว่าเหรียญใดหนักกว่า

ตาชั่งแบบใหม่ของอามินานั้นซับซ้อนกว่าตาชั่งทั่วไป โดยตาชั่งจะมีสี่จาน โดยเรียกเป็นจาน A, จาน B, จาน C และจาน D ตาชั่งดังกล่าวสามารถเลือกใช้งานได้ทั้งหมดสี่รูปแบบ ซึ่งแต่ละรูปแบบจะตอบคำถามที่แตกต่างกันของเหรียญที่นำไปชั่ง ในการใช้งาน อามินาจะต้องวางเหรียญลงในจาน A, จาน B และจาน C จานละหนึ่งเหรียญ นอกจากนี้ ในรูปแบบการใช้งานที่สี่ เธอจะต้องวางเหรียญอีกหนึ่งเหรียญลงในจาน D

ตาชั่งจะตอบคำถามสี่คำถาม ซึ่งจะกำหนดตามรูปแบบการใช้งานทั้งสี่แบบ ดังนี้

1. เหรียญใดในจาน A, จาน B, และจาน C มีน้ำหนักมากที่สุด?
2. เหรียญใดในจาน A, จาน B, และจาน C มีน้ำหนักน้อยที่สุด?
3. เหรียญใดในจาน A, จาน B, และจาน C มีน้ำหนักที่เป็นน้ำหนักมัธยฐาน? (นั่นคือเหรียญที่ไม่ใช่เหรียญที่หนักที่สุด และไม่ใช่เหรียญที่เบาที่สุด)
4. จากเหรียญในจาน A, จาน B, และจาน C พิจารณาเฉพาะเหรียญที่หนักกว่าเหรียญในจาน D ถ้ามีเหรียญตามเงื่อนไขดังกล่าว จะตอบคำถามว่า เหรียญที่ตรงเงื่อนไขในจานใดมีน้ำหนักน้อยที่สุด? อย่างไรก็ตาม ถ้าไม่มีเหรียญที่ตรงตามเงื่อนไข จะตอบคำถามว่า เหรียญใดในจาน A, จาน B, หรือจาน C มีน้ำหนักที่น้อยที่สุด?

งานของคุณ

เขียนโปรแกรมสำหรับเรียงเหรียญทั้ง 6 ของอามินาตามน้ำหนัก โปรแกรมดังกล่าวสามารถเรียกใช้งานตาชั่งเพื่อเปรียบเทียบน้ำหนักของเหรียญ โปรแกรมของคุณจะได้รับข้อมูลชุดทดสอบจำนวน

หนึ่งเพื่อแก้ปัญหา แต่ละข้อมูลทดสอบจะเป็นเหรียญชุดใหม่ 6 เหรียญ

โปรแกรมของคุณต้องมีฟังก์ชัน `init` และ `orderCoins` ระหว่างการทำงานแต่ละครั้งของโปรแกรมของคุณ เกรดเดอร์จะเรียก `init` หนึ่งครั้ง การเรียกครั้งนี้จะระบุจำนวนข้อมูลทดสอบและเปิดโอกาสให้คุณกำหนดค่าเริ่มต้นของตัวแปรต่าง ๆ จากนั้นเกรดเดอร์จะเรียก `orderCoins()` หนึ่งครั้งต่อหนึ่งข้อมูลทดสอบ

- `init(T)`
  - `T`: จำนวนข้อมูลทดสอบที่โปรแกรมของคุณจะต้องแก้ในการทำงานครั้งนี้ `T` จะเป็นจำนวนเต็มจากช่วง $1, \ldots, 18$
  - ฟังก์ชันนี้ไม่ต้องคืนค่า (ไม่มี return value)
- `orderCoins()`
  - ฟังก์ชันนี้จะถูกเรียกใช้หนึ่งครั้งต่อหนึ่งข้อมูลทดสอบ
  - ฟังก์ชันนี้จะต้องคำนวณหาลำดับที่ถูกต้องของเหรียญของอามินา โดยการเรียกใช้ฟังก์ชันของเกรดเดอร์ต่อไปนี้ `getHeaviest()`, `getLightest()`, `getMedian()`, และ/หรือ `getNextLightest()`
  - เมื่อฟังก์ชันทราบลำดับที่ถูกต้องแล้ว ฟังก์ชันจะต้องรายงานคำตอบโดยเรียกใช้เกรดเดอร์ฟังก์ชัน `answer()`
  - หลังจากเรียกฟังก์ชัน `answer()` แล้ว ฟังก์ชัน `orderCoins()` จะต้องจบการทำงาน ฟังก์ชันนี้ไม่ต้องคืนค่า (ไม่มี return value)

คุณสามารถใช้ฟังก์ชันจากเกรดเดอร์ต่อไปนี้ได้

- `answer(W)` — โปรแกรมของคุณจะต้องใช้ฟังก์ชันนี้เพื่อรายงานคำตอบที่คุณหาได้
  - `W`: อาร์เรย์ความยาว 6 ที่ระบุลำดับที่ถูกต้องของเหรียญ `W[0]` ไปจนถึง `W[5]` จะต้องระบุหมายเลขของเหรียญ (นั่นคือจำนวนเต็มจาก 1 ถึง 6) เรียงตามลำดับจากน้อยไปหามาก
  - โปรแกรมของคุณจะต้องเรียกฟังก์ชันนี้จากฟังก์ชัน `orderCoins()` หนึ่งครั้งต่อหนึ่งข้อมูลทดสอบ
  - ฟังก์ชันนี้ไม่คืนค่า (ไม่มี return value)

- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — ฟังก์ชันนี้จะสอดคล้องกับรูปแบบการใช้งานที่ 1, 2, และ 3 ของตาชั่งของอามินาตามลำดับ
  - `A`, `B`, `C`: เหรียญที่ถูกวางลงในจาน A, จาน B, และจาน C ตามลำดับ ค่าในตัวแปร `A`, `B`, และ `C` จะต้องระบุจำนวนเต็มที่แตกต่างกันและมีค่าระหว่าง 1 ถึง 6 (รวม 1 และ 6 ด้วย)
  - แต่ละฟังก์ชันจะคืนค่าใดค่าหนึ่งในตัวแปร `A`, `B`, และ `C` ซึ่งระบุหมายเลขของเหรียญที่เป็นคำตอบของคำถาม ตัวอย่างเช่น `getHeaviest(A, B, C)` คืนหมายเลขของเหรียญที่หนักที่สุดในเหรียญสามเหรียญที่ส่งให้
- `getNextLightest(A, B, C, D)` — ฟังก์ชันนี้สอดคล้องกับรูปแบบการใช้งานที่ 4 ของตาชั่งของอามินา
  - `A`, `B`, `C`, `D`: เหรียญที่ถูกวางลงในจาน A, จาน B, จาน C, และจาน D ตามลำดับ ค่าในตัวแปร `A`, `B`, `C`, และ `D` จะต้องระบุจำนวนเต็มที่แตกต่างกันและมีค่าระหว่าง 1 ถึง 6 (รวม 1 และ 6 ด้วย)
  - แต่ละฟังก์ชันจะคืนค่าใดค่าหนึ่งในตัวแปร `A`, `B`, และ `C` ซึ่งระบุหมายเลขของเหรียญที่มีน้ำหนักน้อยที่สุดในกลุ่มของเหรียญในจาน A, จาน B, และจาน C ที่มีน้ำหนักมากกว่าเหรียญในจาน D หรือในกรณีที่ไม่มีเหรียญใดที่หนักกว่าเหรียญในจาน D, เหรียญที่คืนจะเป็นเหรียญที่เบาที่สุดในบรรดาเหรียญทั้งสามในจาน A, จาน B, และจาน C

การให้คะแนน

โจทย์ข้อนี้ไม่มีปัญหาย่อย อย่างไรก็ตาม คะแนนของคุณจะถูกคำนวณตามจำนวนครั้งที่คุณใช้ในการชั่งเหรียญ (นั่นคือ จำนวนครั้งที่คุณเรียกฟังก์ชันของเกรดเดอร์ `getLightest()`, `getHeaviest()`, `getMedian()` และ/หรือ `getNextLightest()`)

โปรแกรมจะถูกเรียกให้ทำงานหลายครั้งกับข้อมูลทดสอบหลายกรณีต่อการทำงานแต่ละครั้ง ให้ r เป็นจำนวนครั้งที่โปรแกรมของคุณถูกเรียกให้ทำงาน (จำนวน run) ตัวเลขนี้จะไม่เปลี่ยนแปลงและขึ้นกับข้อมูลทดสอบ ถ้าโปรแกรมของคุณไม่สามารถเรียงเหรียญได้ถูกต้องในข้อมูลทดสอบชุดใดชุดหนึ่ง ของการทำงานครั้งใด ๆ คุณจะได้ 0 คะแนน ในกรณีอื่น ๆ แต่ละการทำงาน (แต่ละ run) จะได้รับการพิจารณาคะแนนแยกจากกันตามเงื่อนไขดังต่อไปนี้

ให้ Q เป็นจำนวนที่น้อยที่สุดที่สามารถเรียงลำดับของเหรียญหกเหรียญใด ๆ ก็ได้โดยใช้การชั่ง Q ครั้ง ด้วยตาชั่งของอามินา เพื่อให้โจทย์ข้อนี้ท้าทาย เราจะไม่เปิดเผยค่าของ Q ณ ที่นี้

สมมติว่าจำนวนครั้งที่ใช้ในการชั่งที่มากที่สุดในทุกข้อมูลทดสอบของทุกการทำงานคือ $Q + y$ สำหรับจำนวนเต็ม y บางจำนวน จากนั้นให้พิจารณาการทำงาน (run) หนึ่ง ๆ ของโปรแกรมของคุณ ให้จำนวนครั้งของการชั่งที่โปรแกรมใช้ในการเรียงข้อมูลทดสอบ T กรณีในการทำงาน (run) นี้เป็น $Q + x$ สำหรับบางจำนวนเต็มไม่เป็นลบ x (ถ้าคุณใช้น้อยกว่า Q ในทุก ๆ ข้อมูลทดสอบ เราจะให้ $x = 0$) คะแนนของคุณในการทำงานนี้ (run) จะมีค่าเท่ากับ

$$\frac{100}{r((x+y)/5+1)}$$

*ปัดลง*ให้เหลือเป็นจำนวนทศนิยมสองตำแหน่ง

โดยเฉพาะอย่างยิ่ง ถ้าโปรแกรมของคุณใช้การชั่งไม่เกิน Q ในทุก ๆ ข้อมูลทดสอบในทุก ๆ การทำงาน คุณจะได้คะแนน 100 คะแนน

ตัวอย่าง

สมมติว่าเหรียญเรียงลำดับได้เป็น 3 4 6 2 1 5 จากเบาสุดไปยังหนักที่สุด

Function call	Returns	Explanation
getMedian(4, 5, 6)	6	เหรียญ 6 เป็นเหรียญมัธยฐานของเหรียญ 4, 5, และ 6
getHeaviest(3, 1, 2)	1	เหรียญ 1 เป็นเหรียญที่หนักที่สุดใน 1, 2, และ 3
getNextLightest(2, 3, 4, 5)	3	เนื่องจากเหรียญ 2, 3, และ 4 นั้นเบากว่าเหรียญ 5, ดังนั้น หมายเลขของเหรียญที่เบาที่สุดในกลุ่มนี้ (3) จึงคืนกลับมา
getNextLightest(1, 6, 3, 4)	6	เนื่องจากเหรียญ 1 และ 6 นั้นหนักกว่าเหรียญ 4 ระหว่างเหรียญ 1 และ 6, เหรียญ 6 เป็นเหรียญที่เบาที่สุด
getHeaviest(3, 5, 6)	5	เหรียญ 5 เป็นเหรียญที่หนักที่สุดในเหรียญ 3, 5 และ 6
getMedian(1, 5, 6)	1	เหรียญ 1 เป็นเหรียญมัธยฐานของเหรียญ 1, 5 และ 6.
getMedian(2, 4, 6)	6	เหรียญ 6 เป็นเหรียญมัธยฐานของเหรียญ 2, 4 และ 6.
answer([3, 4, 6, 2, 1, 5])		โปรแกรมพบคำตอบที่ถูกต้องของข้อมูลชุดทดสอบนี้

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างอ่านข้อมูลนำเข้าในรูปแบบต่อไปนี้:

- บรรทัดที่ 1: T — จำนวนของข้อมูลทดสอบ
- แต่ละบรรทัดตั้งแต่บรรทัดที่ 2 ถึง $T + 1$: ลำดับของตัวเลขที่แตกต่างกัน 6 ตัว ที่มีค่าระหว่าง

1 ถึง 6 ที่ระบุลำดับของเหรียญเรียงจากเหรียญที่หนักน้อยที่สุดไปยังเหรียญที่หนักมากที่สุด

ตัวอย่างเช่น ข้อมูลนำเข้าที่ประกอบด้วยข้อมูลทดสอบสองชุดที่เหรียญเรียงลำดับเป็น 1 2 3 4 5 6 และ 3 4 6 2 1 5 จะเป็นดังนี้:

```
2
1 2 3 4 5 6
3 4 6 2 1 5
```

เกรดเดอร์ตัวอย่างพิมพ์อาร์เรย์ที่ถูกส่งให้กับฟังก์ชัน `answer()`