การแข่งขันคอมพิวเตอร์โอลิมปิกระดับชาติ ครั้งที่ 21
ณ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ข้อสอบข้อที่ 3 จากทั้งหมด 3 ข้อ
วันพุธที่ 14 พฤษภาคม 2568 เวลา 08.30 – 13.30 น.

รถไฟตู้เสบียง (Dining Car)

ในยุคสมัยที่การเดินทางด้วยรถไฟยังคงเป็นเสน่ห์ของการผจญภัยและการค้นพบ มีตำนานเล่าขานถึงขบวนรถไฟสุดหรู "โอเรียนท์สยาม" ที่ทอดตัวยาวไปตามรางเหล็กอันคดเคี้ยว ผ่านผืนป่าลึกลับ หุบเขาลึก และเมืองโบราณที่ซ่อนตัวอยู่ริมทางรถไฟ ขบวนรถไฟนี้มิใช่เพียงพาหนะธรรมดา หากแต่เป็นดั่งโรงแรมเคลื่อนที่

ขบวนรถไฟโอเรียนท์สยาม ประกอบไปด้วยตู้โดยสารจำนวน N ตู้ ซึ่งถูกจัดเรียงตามลำดับในขบวน ตั้งแต่ตู้หมายเลข 1 ที่อยู่หัวขบวนจนถึงตู้หมายเลข N ที่อยู่ท้ายขบวน

เพื่อเติมเต็มประสบการณ์การเดินทางอันแสนพิเศษ ขบวนรถไฟโอเรียนท์สยาม ยังมี “ตู้เสบียง” พิเศษ<u>สองตู้</u> คือตู้หมายเลข A และตู้หมายเลข B เมื่อ $1 \leq A < B \leq N$ ที่ให้บริการอาหารตลอดการเดินทาง คุณในฐานะผู้โดยสารไม่ทราบหมายเลขของตู้เสบียงทั้งสองนี้และไม่สามารถสอบถามหมายเลขตู้ได้โดยตรง อย่างไรก็ตามคุณสามารถเลือกหมายเลขตู้ P และ Q สองหมายเลขใด ๆ และ “ถาม” พนักงานรถไฟว่า**เวลาเดินทาง**<u>น้อยที่สุด</u>จากตู้หมายเลข P ไปยังตู้เสบียงใด ๆ นั้นมีค่า น้อยกว่า เท่ากับ หรือ มากกว่า **เวลาเดินทาง**<u>น้อยที่สุด</u>จากตู้หมายเลข Q ไปยังตู้เสบียงใด ๆ

กำหนดให้ **เวลาเดินทาง**ระหว่างสองตู้โดยสารคำนวณได้จาก “ค่าสัมบูรณ์ของผลต่างของหมายเลขตู้” กล่าวคือ **เวลาเดินทาง**ระหว่างตู้หมายเลข i ไปยังตู้หมายเลข j เท่ากับ $|i - j|$ หน่วย

คุณต้องการหาหมายเลขของตู้เสบียงทั้ง 2 ตู้ผ่านการ “ถาม” ในรูปแบบที่กล่าวมาแล้ว นอกจากนั้นแล้วคุณต้องการ “ถาม” เป็นจำนวนน้อย ๆ ครั้ง

งานของคุณ (Your Task)

จงเขียนฟังก์ชันเพื่อหาหมายเลขตู้ A และ B ของตู้เสบียง

ข้อนี้มีรูปแบบการเขียนโปรแกรมที่แตกต่างจากรูปแบบปรกติ **และมีการเรียกฟังก์ชันที่ระบบเตรียมไว้ให้** โปรดดูส่วนท้ายของโจทย์ข้อนี้เกี่ยวกับวิธีการใช้งาน

รายละเอียดการเขียนโปรแกรม

ผู้เข้าแข่งขันจะต้องเขียนฟังก์ชันต่อไปนี้

```
std::pair<int,int> locate_dining_cars (int N)
```

โดยที่พารามิเตอร์คือ

- N คือ จำนวนตู้โดยสารทั้งหมดของขบวนรถไฟโอเรียนท์สยาม

ฟังก์ชันนี้จะต้องคืนค่าคู่ A และ B ในรูปของ `std::pair<int, int>` ซึ่งเป็นหมายเลขของตู้เสบียงทั้งสองตู้ โดยที่ $A < B$

ฟังก์ชัน `locate_dining_cars` จะถูกเรียกใช้<u>เพียงหนึ่งครั้ง</u>ต่อกรณีทดสอบ

ภายในฟังก์ชัน `locate_dining_cars` โปรแกรมของคุณสามารถเรียกใช้ฟังก์ชัน compare_cars เพื่อ “ถาม” พนักงานรถไฟ

```
int compare_cars(int P, int Q)
```

โดยที่พารามิเตอร์ต่าง ๆ คือ

- P คือ หมายเลขตู้โดยสารตู้แรกที่คุณเลือก $(1 \leq P \leq N)$
- Q คือ หมายเลขตู้โดยสารตู้ที่สองที่คุณเลือก $(1 \leq Q \leq N)$

ฟังก์ชันนี้จะคืนค่าเป็นจำนวนเต็มต่าง ๆ ดังนี้

- คืนค่า -1 ถ้า**เวลาเดินทาง**<u>น้อยที่สุด</u>จากตู้หมายเลข P ไปยังตู้เสบียงใด ๆ นั้นมีค่า<u>น้อยกว่า</u> **เวลาเดินทาง**<u>น้อยที่สุด</u>จากตู้หมายเลข Q ไปยังตู้เสบียงใด ๆ
- คืนค่า 0 ถ้า**เวลาเดินทาง**<u>น้อยที่สุด</u>จากตู้หมายเลข P ไปยังตู้เสบียงใด ๆ นั้นมีค่า<u>เท่ากับ</u> **เวลาเดินทาง**<u>น้อยที่สุด</u>จากตู้หมายเลข Q ไปยังตู้เสบียงใด ๆ
- คืนค่า 1 ถ้า**เวลาเดินทาง**<u>น้อยที่สุด</u>จากตู้หมายเลข P ไปยังตู้เสบียงใด ๆ นั้นมีค่า<u>มากกว่า</u> **เวลาเดินทาง**<u>น้อยที่สุด</u>จากตู้หมายเลข Q ไปยังตู้เสบียงใด ๆ

คุณสามารถเรียกใช้ฟังก์ชัน `compare_cars` ได้<u>ไม่เกิน 90 ครั้ง</u> สำหรับการเรียกใช้ฟังก์ชัน `locate_dining_cars` หนึ่งครั้ง หากเรียกฟังก์ชัน compare_cars มากกว่า 90 ครั้ง โปรแกรมจะหยุดทำงานและได้ผลการตรวจเป็นผิดทันที

ขอบเขตข้อมูล

- $2 \leq N \leq 10^9$
- $1 \leq A < B \leq N$

ตัวอย่าง

สมมติว่าขบวนรถไฟโอเรียนท์สยาม มี 5 ตู้ และตู้เสบียง คือตู้หมายเลข 1 และ 4 $(A = 1, B = 4)$ ทำให้โปรแกรมของคุณเรียกฟังก์ชัน `locate_dining_cars (5)`

- ถ้าคุณ “ถาม” พนักงานรถไฟ โดยเลือกตู้หมายเลข 1 และ 3 ซึ่งคือการเรียกฟังก์ชัน

  `compare_cars(1,3)`

  - **เวลาเดินทาง**น้อยที่สุดจากตู้หมายเลข 1 ไปยังตู้เสบียง (ตู้หมายเลข 1) คือ |1 - 1| = 0 หน่วย
  - **เวลาเดินทาง**น้อยที่สุดจากตู้หมายเลข 3 ไปยังตู้เสบียง (ตู้หมายเลข 4) คือ |3 - 4| = 1 หน่วย

  ในกรณีนี้ เวลาเดินทางน้อยที่สุดของตู้หมายเลข 1 น้อยกว่า เวลาเดินทางน้อยที่สุดของตู้หมายเลข 3 ฟังก์ชัน `compare_cars(1,3)` จะคืนค่า -1
- ถ้าคุณ “ถาม” พนักงานรถไฟเกี่ยวกับตู้หมายเลข 3 และ 5 ซึ่งคือการเรียกฟังก์ชัน

  `compare_cars(3,5)`

  - **เวลาเดินทาง**น้อยที่สุดจากตู้หมายเลข 3 ไปยังตู้เสบียง (ตู้หมายเลข 4) คือ |3 - 4| = 1 หน่วย
  - **เวลาเดินทาง**น้อยที่สุดจากตู้หมายเลข 5 ไปยังตู้เสบียง (ตู้หมายเลข 4) คือ |5 - 4| = 1 หน่วย

  ในกรณีนี้ เวลาเดินทางน้อยที่สุดของตู้หมายเลข 3 เท่ากับ เวลาเดินทางน้อยที่สุดของตู้หมายเลข 5 ฟังก์ชัน `compare_cars(3,5)` จะคืนค่า 0

ข้อกำหนด

หัวข้อ	เงื่อนไข
ระยะเวลาสูงสุดที่ใช้ในการประมวลผล	1 วินาที
หน่วยความจำสูงสุดที่ใช้ในการประมวลผล	1024 MB
คะแนนสูงสุดของโจทย์	100 คะแนน

ข้อมูลเพิ่มเติมเกี่ยวกับชุดทดสอบ

<table>
<tr><th>กลุ่มชุดทดสอบที่</th><th>อัตราส่วนคะแนน</th><th>เงื่อนไข</th></tr>
<tr><td>1</td><td>3%</td><td>$N \leq 10$</td></tr>
<tr><td>2</td><td>7%</td><td>$N \leq 50$</td></tr>
<tr><td>3</td><td>12%</td><td>$N \leq 10^6$ และตู้เสบียงทั้งสองตู้อยู่ติดกัน ($B = A + 1$)</td></tr>
<tr><td>4</td><td>17%</td><td>$N \leq 10^6$ และ $A = 1$ (ตู้เสบียงตู้แรกอยู่ที่หัวขบวน)</td></tr>
<tr><td>5</td><td>21%</td><td>$N \leq 10^6$</td></tr>
<tr><td>6</td><td>40%</td><td>ไม่มีเงื่อนไขเพิ่มเติม
เกณฑ์การให้คะแนน พิจารณาจาก<u>จำนวนครั้งที่มากที่สุด</u> (q_{max}) ในการเรียกใช้ฟังก์ชัน <code>compare_cars</code> จากทุกกรณีทดสอบ
<table>
<tr><th>ค่า q_{max}</th><th>อัตราส่วนคะแนน</th></tr>
<tr><td>$q_{max} \leq 60$</td><td>40% จากทั้งหมด</td></tr>
<tr><td>$60 < q_{max} \leq 90$</td><td>$40 \times \left(\frac{90 - q_{max}}{30}\right)^2$% จากทั้งหมด</td></tr>
<tr><td>$90 < q_{max}$</td><td>0% จากทั้งหมด</td></tr>
</table>
</td></tr>
</table>

คำอธิบายไฟล์เกรดเดอร์ (grader.cpp)

ไฟล์เกรดเดอร์จะรับข้อมูลนำเข้าหนึ่งบรรทัดในรูปแบบต่อไปนี้

- บรรทัดที่ 1: รับค่า N , A และ B

หลังจากนั้นไฟล์เกรดเดอร์จะเรียกฟังก์ชัน `locate_dining_cars` ของผู้เข้าแข่งขัน แล้วแสดงผลลัพธ์ที่ได้จากฟังก์ชัน และจำนวนครั้งที่ฟังก์ชัน `compare_cars` ถูกเรียกออกทางหน้าจอ

คำแนะนำในการเขียนโปรแกรม

สำหรับข้อนี้ วิธีการเขียนโปรแกรมและส่งโปรแกรมเข้าสู่ระบบเกรดเดอร์ (Grader) จะเป็นรูปแบบที่แตกต่างจากรูปแบบทั่วไปของการเขียนโปรแกรม ที่ผู้เข้าแข่งขันจะต้องเขียนโปรแกรมในส่วนรับข้อมูลนำเข้า และส่วนแสดงผล โดยในข้อนี้ ผู้เข้าแข่งขันจะต้องเขียนโปรแกรมเฉพาะส่วนการคำนวณที่จำเป็นลงในฟังก์ชันที่กำหนดให้ โดยข้อมูลนำเข้าต่าง ๆ จะสามารถใช้ได้ผ่านพารามิเตอร์ของฟังก์ชัน และผลการ

คำนวณจะต้องส่งกลับผ่านฟังก์ชันด้วยเช่นกัน โดยที่ผู้เข้าแข่งขันไม่ต้องเขียนโปรแกรมในส่วนรับข้อมูลนำเข้า และไม่ต้องเขียนส่วนแสดงผล

ผู้เข้าแข่งขันจะได้รับไฟล์ต่าง ๆ จำนวน 5 ไฟล์ดังนี้ ผู้เข้าแข่งขันจะต้องแก้ไขไฟล์เดียวตามที่โจทย์กำหนด และส่งไฟล์ดังกล่าวเข้าสู่ระบบ

- grader.cpp เป็นไฟล์หลักที่ทำหน้าที่รับข้อมูลนำเข้าและส่งออก
- dining_car.cpp เป็นไฟล์ที่ผู้เข้าแข่งขันจะต้องเขียนโปรแกรมคำนวณต่าง ๆ และเป็นไฟล์ที่ผู้เข้าแข่งขันส่งเข้าสู่ระบบ (ในไฟล์ dining_car.cpp ห้ามมีฟังก์ชัน main() แต่มีฟังก์ชันอื่น ๆ ได้)
- dining_car.h เป็นไฟล์ที่ระบุชื่อฟังก์ชัน (ผู้เข้าแข่งขันไม่ต้องแก้ไขไฟล์นี้)
- run_cpp.sh เป็นไฟล์ที่ผู้เข้าแข่งขันสามารถเรียกใช้เพื่อทำการรันโปรแกรมทั้งหมดได้
- compile_cpp.sh เป็นไฟล์ที่ผู้เข้าแข่งขันสามารถเรียกใช้เพื่อทำการคอมไพล์โปรแกรมได้

ผู้เข้าแข่งขันจะได้รับไฟล์ .zip ซึ่งสามารถดาวน์โหลดจากระบบโดยในไฟล์ดังกล่าวเมื่อขยาย (extract) ออกมาแล้วจะมีไฟล์ทั้งหมดอยู่แล้ว ผู้เข้าแข่งขันควรใช้ text editor หรือ IDE ในการแก้ไขไฟล์ dining_car.cpp และทำการคอมไพล์หรือรันโปรแกรมผ่านการเรียกใช้ฟังก์ชัน compile_cpp.sh หรือ run_cpp.sh

ผู้เข้าแข่งขันสามารถเรียกไฟล์ run_cpp.sh เพื่อทำการรันโปรแกรม โปรแกรมจะทำงานตามที่ได้ระบุไว้ในไฟล์ grader.cpp ซึ่งหัวข้อ “**คำอธิบายไฟล์เกรดเดอร์ (grader.cpp)**” ได้อธิบายการทำงานของไฟล์ดังกล่าวไว้แล้ว