Queue Merge Greater (1 sec, 512 mb)

จงเพิ่มบริการ `void CP::queue::merge_greater(CP::queue<T> &other)` ให้กับ `CP::queue` โดยฟังก์ชันนี้จะรับ `CP::queue` อีกตัวเข้ามาและจัดให้ด้านหน้าของคิวทั้งสองตรงกัน จากนั้นจะเทียบข้อมูล element ที่ตำแหน่งตรงกันของทั้งสองคิว ("ตำแหน่ง" หมายถึงตำแหน่งนับจากหัวคิว เช่น ตำแหน่งที่ `0` คือข้อมูลของหัวคิว และตำแหน่งที่ `1` คือข้อมูลถัดจากหัวคิว) โดยหากเทียบข้อมูลตำแหน่งที่ `k` ของทั้งสองคิว แล้วปรากฏว่า**ข้อมูลของคิว other มีค่ามากกว่าข้อมูลของคิวที่เรียกใช้ฟังก์ชัน** จะย้ายข้อมูลที่ตำแหน่ง `k` ของคิว other มาแทรกที่ตำแหน่ง `k + 1` ของคิวที่เรียกใช้ฟังก์ชัน

ตัวอย่างเช่น หากให้ `queue A` เรามีข้อมูลภายในเป็น `{1,5,7,3,10,9,4,3,-1,0}` โดยที่ซ้ายสุดคือหน้าคิว (กล่าวคือ `A.front()` จะให้ค่า `1`) และให้ `queue B` มีข้อมูลภายในเป็น `{-5,8,12,2,11}` แล้วเราเรียก `A.merge_greater(B)` จะทำให้ `queueA` มีข้อมูลภายในกลายเป็น `{1,5,8,7,12,3,10,11,9,4,3,-1,0}` ส่วน `queue B` จะเหลือข้อมูลภายในเป็น `{-5,2}` **โดยถ้า `queue A` และ `B` เป็น queue เดียวกัน จะไม่เกิดอะไรขึ้น**

เพื่อความง่ายในการทำข้อสอบและ**คนออกโจทย์เป็นคนใจดีมาก ๆ :)** รับประกันว่า `queue B` **จะมีขนาดไม่เกินครึ่งหนึ่งของ** `queue A` (กล่าวอีกนัยคือ `B.size() ≤ A.size() / 2`) โดย Address ของ `queue A` กับ `queue B` และ `mData` ของแต่ละคิว**ต้องไม่เปลี่ยน (ยกเว้น `mData` ของ A จะเปลี่ยน Address ได้เมื่อมีการขยายความจุเท่านั้น)**

ข้อบังคับ

- โจทย์ข้อนี้จะมีไฟล์ตั้งต้นมาให้ซึ่งประกอบด้วยไฟล์ queue.h, main.cpp และ student.h อยู่ ให้เขียน code เพิ่มเติมลงในไฟล์ student.h เท่านั้น และการส่งไฟล์เข้าสู่ระบบ grader ให้ **ส่งเฉพาะไฟล์ student.h เท่านั้น**
    - ไฟล์ student.h จะต้อง**ไม่ทำการอ่านเขียนข้อมูลใด ๆ ไปยังหน้าจอหรือคีย์บอร์ดหรือไฟล์ใด ๆ**
- หากใช้ VS Code ให้ทำการ compile ที่ไฟล์ main.cpp

  **** main ที่ใช้จริงใน grader นั้นจะแตกต่างจาก main ที่ได้รับในไฟล์เริ่มต้น แต่จะทดสอบในลักษณะเดียวกัน ****

คำอธิบายฟังก์ชัน main

`main()` จะทดลองใช้งาน `CP::queue` โดย main จะสร้าง `CP::queue<int>` ไว้ 2 ตัวชื่อ `A` และ `B` และอ่านคำสั่งทีละบรรทัดดังนี้ (`m` และ `k` เป็นแค่ตัวอักษร `'A'` หรือ `'B'` ส่วน `x` เป็นจำนวนเต็มที่อยู่ในของเขต `int`)

บรรทัดแรก จำนวนเต็ม `A` และ `B` แทนจำนวนข้อมูลใน `queue A` และ `queue B` โดยถ้า `A` หรือ `B` เป็น `0` จะไม่มีบรรทัดต่อมาที่สอดคล้องกับตัวแปรนั้น (ดูตัวอย่างที่ 2) $(1 \le A \le 5x10^5, 1 \le B \le (A/2))$

บรรทัดที่สอง จำนวนเต็ม `A` ตัวแทนข้อมูลของ `queue A` โดยข้อมูลซ้ายสุดคือหน้าคิว

บรรทัดที่สาม จำนวนเต็ม `B` ตัวแทนข้อมูลของ `queue B` โดยข้อมูลซ้ายสุดคือหน้าคิว

หลายบรรทัดต่อมา เริ่มต้นด้วย string ระบุคำสั่ง และอาจตามด้วยค่าต่างๆ ที่คำสั่งนั้นต้องการ โดยมีรูปแบบดังนี้

คำสั่ง	คำอธิบาย
`p m`	พิมพ์ `mSize`, `front()` และ `back()` ของ `queue m` (จะไม่สั่งตอน queue ว่างแน่นอน)
`e m x`	push x เข้าท้าย queue m (เรียก `m.push(x)`)
`d m`	เรียก `m.pop()`
`g m k`	เรียก `m.merge_greater(k)` (`queue k` merge ลง `queue m`)
`q`	จบการทำงาน

ตัวอย่างการทำงานของ main

```
10 5
1 5 7 3 10 9 4 3 -1 0
-5 8 12 2 11
p A
g A B
p B
d A
d A
p A
q
```

```
10 1 0
2 -5 2
11 8 0
```

```
5 0
5 7 -9 3 4
p A
g A B
p A
e B 8
d A
g A B
d A
p A
q
```

```
5 5 4
5 5 4
4 8 4
```

```
4 2
6 9 -10 2
5 10
e B 8
e B 12
d A
d A
g B A
p A
p B
q
```

```
2 -10 2
4 5 12
```

ข้อมูลชุดทดสอบ

5 % มีการเรียกใช้ `A.merge_greater(A)` เพียงอย่างเดียว (merge กับตัวเอง)

60 % $A \leq 1000$

35% $B \leq A / 8$

ข้อแนะนำ

- โจทย์ข้อนี้สามารถผ่านได้ด้วย time complexity และ space complexity เป็น $O(n)$ ไม่จำเป็นต้อง optimize มาก
- สังเกตว่าไม่มีการพิมพ์ `mFront` และ `mCap` จึงสามารถเปลี่ยนค่าอย่างไรก็ได้ แค่ให้ข้อมูลเรียงถูกต้อง

Author: enderpalm