List Implant (1 sec, 512 mb)

จงเพิ่มบริการ `void CP::list::implant(CP::list<T> &l2,int pos1,int pos2,int count)` ให้กับ `CP::list` ซึ่งเป็น Circular Doubly Linked List โดยฟังก์ชันนี้จะรับ `CP::list` อีกหนึ่งตัวเข้ามาชื่อ `l2` ฟังก์ชันนี้จะทำการย้ายข้อมูลบางส่วนจาก `l2` ไปยัง `list` ตั้งต้น โดยมีเงื่อนไขดังนี้

- `pos1` คือ ตำแหน่งในลิสต์ปัจจุบัน (this) ที่จะทำการแทรกโหนดของอีกลิสต์ หลังตำแหน่งนี้
- `pos2` คือ ตำแหน่งเริ่มต้นในลิสต์ `l2` ที่จะเริ่มตัดโหนดออกมา
- count คือจำนวนโหนดที่ต้องการตัดออกจาก `l2` และแทรกเข้าไปในลิสต์ปัจจุบัน
- การนับตำแหน่งเป็นแบบ 0-based Index
- รับรองว่า `pos1, pos2` จะมีค่าไม่เกิน `mSize - 1`
- `pos1, pos2` จะมีค่าไม่น้อยกว่า `0` และ `pos2 + count` จะมีค่าไม่เกิน `mSize`

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

คำอธิบายฟังก์ชัน main

`main()` จะทดลองใช้งานฟังก์ชัน โดย main จะสร้าง `CP::list<int>` 2 ตัวชื่อ `l1` และ `l2` และอ่านคำสั่งดังนี้

บรรทัดแรก จำนวนเต็ม A และ B แทนจำนวนข้อมูลใน `l1` และ `l2` ($1 \le A, B \le 1 \times 10^5$)

บรรทัดที่สอง จำนวนเต็ม A ตัวแทนข้อมูลของ `l1` โดยข้อมูลแรกสุดคือข้อมูลอันดับแรกสุดของ `l1`

บรรทัดที่สาม จำนวนเต็ม B ตัวแทนข้อมูลของ `l2` โดยข้อมูลแรกสุดคือข้อมูลอันดับแรกสุดของ `l2`

บรรทัดที่สี่ จำนวนเต็ม `pos1, pos2, count`

จากนั้นโปรแกรมจะสั่งคำสั่ง `l1.implant(l2, pos1, pos2, count)` และจะส่งออกข้อมูลเพียงบรรทัดเดียวคือข้อมูลทั้งหมดใน `l1` หลังจากทำการสั่ง `implant` แล้ว

เพิ่มเติม

- ฟังก์ชัน implant ทำการย้ายข้อมูลบางส่วนใน `l2` ไปยัง `l1` ดังนั้นข้อมูลส่วนที่ถูกย้ายนั้นจะไม่อยู่ใน `l2` อีกต่อไป
- main.cpp ที่น้องได้รับจะไม่เหมือนกันกับในระบบ grader โดย main.cpp ที่น้องได้รับจะตรวจแค่ข้อมูลใน `l1`
- แต่ว่า main.cpp ใน grader จะตรวจรายละเอียดทั้งหมดของ `l1` และ `l2`
- จุดประสงค์ของข้อสอบ Linked List คือน้องต้องสามารถแก้ไข Pointer ให้ได้ ดังนั้น**ไม่อนุญาตให้ใช้ฟังก์ชัน insert และ erase อย่างไรก็ตาม 30% แรกของข้อมูลชุดทดสอบจะอนุญาตให้ใช้ได้**
- กฎเกณฑ์ที่กล่าวมานี้เคยถูกใช้ในข้อสอบจริง

ตัวอย่างการทำงานของ main

6 5 1 2 3 4 5 6 -1 -2 -3 -4 -5 2 1 2	1 2 3 -2 -3 4 5 6
6 5 1 2 3 4 5 6 -1 -2 -3 -4 -5 5 0 3	1 2 3 4 5 6 -1 -2 -3
6 5 1 2 3 4 5 6 -1 -2 -3 -4 -5 0 3 2	1 -4 -5 2 3 4 5 6
6 7 1 2 3 4 5 6 67 67 67 67 67 67 67 2 3 0	1 2 3 4 5 6

ข้อมูลชุดทดสอบ

สำหรับ 30% แรกของข้อมูลชุดทดสอบจะอนุญาตให้ใช้ฟังก์ชัน insert และ erase โดยแบ่งเป็นชุดทดสอบย่อย ๆ ได้ ดังต่อไปนี้

5% `l1.size(), l2.size ≤ 20, count = 0`

5% `l1.size(), l2.size ≤ 20, pos1 = 0`

5% `l1.size(), l2.size ≤ 20, pos1 = mSize-1`

15% `l1.size(), l2.size ≤ 20`

70% ไม่มีข้อกำหนดอื่นใด และไม่อนุญาตให้ใช้ฟังก์ชัน insert และ erase

Author: Rathanon Makaramanee