Stack/Queue Append

256MB เวลาไม่เกิน 5 วินาที

ในข้อนี้ให้นิสิตเพิ่มบริการ `appendQueue(CP::queue<T> q)` และ `appendStack(CP::stack<T> s)` ให้กับ `CP::stack` และ `CP::queue` โดยคำสั่งเหล่านี้จะนำข้อมูลที่ส่งเข้ามาเพิ่มให้กับรายการของข้อมูลในปัจจุบัน โดยผลลัพธ์ที่ได้ต้องมีลักษณะดังนี้

- ลำดับในการข้อมูลออกของข้อมูลเดิมไม่เปลี่ยน
- ข้อมูลใหม่ที่นำเข้ามาต้องมีลำดับในการนำข้อมูลออกหลังข้อมูลเดิมเสมอ
- ลำดับในการนำข้อมูลออกของข้อมูลใหม่ต้องเหมือนกับลำดับของโครงสร้างข้อมูลก่อนที่จะนำเข้ามา
- สามารถแก้ไขข้อมูลนำเข้าหรือไม่ก็ได้ตามแต่นิสิตสะดวก

ตัวอย่างที่	ข้อมูลเริ่มต้น	ผลที่ได้
1	`S1` เก็บ `<1,2,3,4,5>` (`5` อยู่บนสุด) และ `S2` เก็บ `<11,12>` (`12` อยู่บนสุด)	`S1.appendStack(S2)` ทำให้ `S1` เก็บ `<11,12,1,2,3,4,5>`
2	`Q1` เก็บ `<6,7,8,9>` (`6` อยู่หน้าสุด) และ `Q2` เก็บ `<16,17>` (`16` อยู่หน้าสุด)	`Q1.appendQueue(Q2)` ทำให้ `Q1` เก็บ `<6,7,8,9,16,17>`
3	`S1` เก็บ `<1,2,3,4,5>` (`5` อยู่บนสุด) และ `Q1` เก็บ `<6,7,8,9>` (`6` อยู่หน้าสุด)	`S1.appendQueue(Q1)` ทำให้ `S1` เก็บ `<9,8,7,6,1,2,3,4,5>`
4	`Q1` เก็บ `<6,7,8,9>` (`6` อยู่หน้าสุด) และ `S1` เก็บ `<1,2,3,4,5>` (`5` อยู่บนสุด)	`Q1.appendStack(S1)` ทำให้ `Q1` เก็บ `<6,7,8,9,5,4,3,2,1>`

โดยไฟล์ `main.cpp` ที่ให้มาแสดงตัวอย่างการเรียกใช้ฟังก์ชัน (ซึ่งเป็นคนละตัวกับ `main.cpp` ที่จะใช้ทดสอบใน Grader) **ให้นิสิตส่งเฉพาะ `student.h` เท่านั้น** นิสิตสามารถแก้ไข `main.cpp` ได้เพื่อทดสอบฟังก์ชันตามต้องการ แต่ห้ามส่ง `main.cpp` ผลลัพธ์ที่คาดหวังเมื่อรัน `main.cpp` (ที่ให้มา)

```
9 8 7 6 1 2 3 4 5
6 7 8 9 5 4 3 2 1
1 1
1 1
```

หมายเหตุ

1. ชุดทดสอบแต่ละชุดจะทดสอบเฉพาะฟังก์ชันใดฟังก์ชันหนึ่งเท่านั้น และแต่ละอันจะเป็น 25% ของจำนวน test case โดยหากนิสิตทำได้ข้อใดข้อหนึ่งก็ขอให้ส่งมาได้ (แต่ต้อง compile ผ่าน) และรับประกันว่าทั้งโครงสร้างข้อมูลเดิมและข้อมูลที่ส่งมาจะมีข้อมูลไม่เกิน 100000 ตัว
2. `stack.h` และ `queue.h` นั้นได้มีการปรับให้เหมาะสมมีคำสั่งเพิ่มเติมและเป็น friend ซึ่งกันและกันทำให้ stack และ queue สามารถเข้าถึงข้อมูล private ของกันและกันได้ (หากอยากใช้)