Transform Infix to Postfix

Infix คือ นิพจน์ที่เขียนตามหลักคณิตพื้นฐาน เป็นนิพจน์ที่ตัวกระทำ (Operator) จะอยู่ตรงกลางระหว่างตัวกระทำ (Operand) ทั้งสองของ Operator นั้น

Postfix คือ นิพจน์ที่อยู่ในรูปแบบที่คอมพิวเตอร์สามารถปฏิบัติตามได้ง่าย เป็นนิพจน์ที่ตัวกระทำ (Operator) จะอยู่ตามหลัง(ถัดจาก) ตัวกระทำ (Operand) ทั้งสองของ Operator นั้น

ในการบ้านนี้ดาวน์โหลดไฟล์การบ้านได้ที่ https://github.com/CS-CMU/cs251 ข้อ HW04 เราใจดี มีโค้ดแปลง Infix เป็น Postfix ให้ (ศึกษาไว้ก็ดีมั้ง เผื่อออกสอบ) แต่เราดันขาดสิ่งที่สำคัญที่สุดคือคลาส CharStack ที่เป็น Data Structure สำหรับอัลกอริทึมนี้ หน้าที่ของคุณคือเขียน 4 ฟังก์ชันในไฟล์ CharStack.cpp โดยแต่ละฟังก์ชันมีรายละเอียดดังนี้ (ดูสไลด์หรือคลิปเพื่อหาข้อมูลเพิ่มเติมได้)

1. `void push(char new_item)` ใส่ตัวอักษร new_item เข้าไปใน Stack
2. `char pop()` เอาตัวอักษรออกจาก Stack พร้อมคืนค่าตัวอักษรนั้นกลับมา
3. `char top()` คืนค่าตัวอักษรที่อยู่ข้างบนสุดของ Stack (อันนี้แตกต่างจากในสไลด์ที่สอน ให้ return เป็น char เลยไม่ต้องเป็น Pointer/Reference แล้ว)
4. `bool isEmpty()` คืนค่าเป็นจริงถ้า Stack ว่าง คืนค่าเป็นเท็จถ้า Stack ไม่ว่าง

อย่าลืมว่าเราสามารถประกาศตัวแปรหรือฟังก์ชันเพิ่มได้เสมอ แต่เกรดเดอร์จะเรียกใช้แค่ 4 ฟังก์ชันที่กำหนด ส่วน Constructor ก็แล้วแต่การ Implement ของแต่ละคน หากสงสัยสามารถสอบถามได้เสมอ

ข้อมูลเข้าและออกสำหรับไฟล์ main.cpp ที่แจกให้

ใช้ได้แค่เครื่องหมายทางคณิตศาสตร์ +-*/() และ ตัวอักษรภาษาอังกฤษตัวเล็ก a ถึง z เท่านั้น (ข้อมูลเข้าไม่มี space)

ตัวอย่าง ในแต่ละข้อแนะนำให้ลองทำเองด้วยมือประกอบเพื่อความเข้าใจด้วย

ข้อมูลเข้า 1	ข้อมูลเข้า 2	ข้อมูลเข้า 3
a+b	(a+b)	(x+y)*z/b
ข้อมูลออก 1	ข้อมูลออก 2	ข้อมูลออก 3
ab+	ab+	xy+z*b/

ข้อมูลเข้า 4	ข้อมูลเข้า 5	ข้อมูลเข้า 6
((t+u+v/w))	(a+b)*(c/d)+(j-l)	(g+k+x)*y+(r+t)/z
ข้อมูลออก 4	ข้อมูลออก 5	ข้อมูลออก 6
tu+vw/+	ab+cd/*jl-+	gk+x+y*rt+z/+