Assignment 5.2

- หาคำตอบของสมการ infix ที่ตรวจสอบจาก 5.1 แล้วว่าไม่มีข้อผิดพลาด
  - ถ้ามีความผิดพลาดจาก 5.1 ให้แจ้ง error แล้ววนรอบรับข้อมูลใหม่
- ควรแปลงสมการ infix ให้อยู่ในรูปของ postfix ก่อน
  - จองอาร์เรย์/ลิสต์ของสตริง เพื่อใช้เก็บ token ที่ได้จากการวิเคราะห์คำ
  - จองอาร์เรย์ /ลิสต์ของสตริง เพื่อใช้เป็น Operator Stack สำหรับเก็บคำสั่ง
  - จองอาร์เรย์ /ลิสต์ของสตริง เพื่อใช้เป็น postfix queue สำหรับเก็บรูปสมการ postfix
  - สร้างฟังก์ชันวนรอบเพื่อแปลง infix arrayให้อยู่ในรูป postfix array ตามอัลกอริทึม (ต้องตรวจสอบความถูกต้องของ postfix ก่อนทำขั้นตอนถัดไป)
- หาคำตอบของสมการที่อยู่ใน postfix array
  - จองอาร์เรย์ของตัวเลขจำนวนจริง เพื่อใช้เป็น Operand stack
  - วนรอบดึงข้อมูลออกจาก postfix array มาทีละตัว แล้วทำตามอัลกอริทึม

Assignment 5.2

- การแปลงสมการ infix ให้อยู่ในรูป postfix
  1. ใส่ "(" และ ")" คร่อม infix expression //ไม่ใส่ก็ได้ แต่ต้องเช็คstack(4) ตอนหมดข้อมูล
  2. ดึงข้อมูล (TOKEN) ออกจาก string ทีละชุด จากซ้ายไปขวา
     - 2.1 ถ้าเป็น "(" ให้ PUSH ใส่ใน STACK (STACK ของ CHAR/STRING)
     - 2.2 ถ้าเป็น ตัวเลข number ให้ส่งไปยัง output //อาจเป็น Queue หรือสตริง
     - 2.3 ถ้าเป็น operator ให้พิจารณาดังนี้
       - 2.3.1 POP operators ทุกตัวที่มีลำดับความสำคัญมากกว่าหรือเท่ากับ operator ที่พบใน 2.3 ส่งไปยัง postfix output
       - 2.3.2 PUSH operator ที่เจอใน 2.3 ใส่เข้าไปใน STACK แทน
     - 2.4 ถ้าเป็น ")" ให้ POP operators ทุกตัวจาก STACK ส่งไปยัง postfix output จนกว่าจะเจอ "(" // pop "(" ออกมา แต่ไม่ต้องส่ง ไป output
  3. ทำซ้ำในข้อ 2 จนกว่าข้อมูลจะหมด // ข้อมูลจะหมดที่ ")" ที่เพิ่มเข้าไป
  4. ถ้ามีข้อมูลเหลืออยู่ใน stack ให้ POP ข้อมูลทั้งหมดออกมาด้วย //กรณีที่ไม่ได้เพิ่ม ")"
- ฟังก์ชันที่แนะนำ
  - ฟังก์ชันเพิ่ม (และ) เข้าไปในสตริง หรืออาร์เรย์ของ token ที่เช็คแล้ว
  - ฟังก์ชัน peek เพื่อดูข้อมูล operator ตัวบนสุดของ stack
  - ฟังก์ชันเพื่อแปลง operator +- , */, ^, !, fn, (,) เป็นเลขตามลำดับความสำคัญ
  - ฟังก์ชันเกี่ยวกับการจัดการ stack ของคำสั่ง(Operator) เช่น
    - pop_operator(token) ทำหน้าที่ดึง token ตัวบนสุดออกจาก stack (2.3.1)
    - push_operator(token) ทำหน้าที่ push token ใส่เข้าใน stack
  - ฟังก์ชันที่ใช้ add token ไปใส่ไว้ใน postfix array

Assignment 5.2

- การหาคำตอบของสมการ postfix
  1. ดึงข้อมูล (TOKEN) ออกจากสมการ(string/queue) ทีละตัวจากซ้ายไปขวา
     - 1.1 ถ้าเป็น ตัวเลข/ค่าคงที่ pi/ตัวแปร ans ให้ PUSH ใส่ใน STACK (ของเลขจำนวนจริง)
     - 1.2 ถ้าเป็น unary operator (Negative, function) ให้ POP ตัวเลข 1 ตัว จาก STACK มาคำนวณตามคำสั่ง แล้วนำคำตอบที่ได้ PUSH ใส่กลับเข้าไปใน STACK

       (กำหนดให้การคำนวณมุมทางเรขาคณิตอยู่ในโหมดของ degree ดังนั้นจะต้องมีการแปลงค่าในคำสั่ง sin, cos, tan, asin, acos, atan ให้ถูกต้องด้วย)
     - 1.3 ถ้าเป็น binary operator (+, -, *, /, ^) ให้ POP ตัวเลขออกจาก STACK 2 ตัวมาคำนวณตามคำสั่ง (POPตัวหลัง operate POPตัวแรก) แล้วนำคำตอบที่ได้ PUSH ใส่กลับเข้าไปใน STACK
  2. ทำซ้ำ ในข้อ 1 จนกว่าข้อมูลจะหมด
  3. POP ตัวเลขจาก STACK (เหลือตัวเดียว) มาเป็นคำตอบ
- ฟังก์ชันที่แนะนำ
  - ฟังก์ชันที่ใช้ตรวจ token ว่าเป็นตัวเลข (ต้องแปลงเป็นจำนวนจริงด้วย)
  - ฟังก์ชันการคำนวณตามรหัสของแต่ละ operator หรือฟังก์ชัน
  - ฟังก์ชันที่แยกรหัสของ operator แต่ละตัว
  - ฟังก์ชัน push ตัวเลขไปยัง stack
  - ฟังก์ชัน pop ตัวเลขที่อยู่ใน stack เพื่อนำมาคำนวณหาคำตอบ

Assignment 5.2 Test Case

```
expression> 1.23  4.56 +
answer> error
expression>sin 90
answer> error
expression> 0/1
answer> 0
expression> 1/sin(0)
answer> error
expression> -2^-2-2
answer>-1.75
13expression> -1+2^3/(4-5*6)+pi
answer> -1.8339
14expression> sqrt(log(10^2)+exp(3))
answer> 4.69953
15expression>sqrt(3^2+4^2)
answer> 5
17expression> ans^-2
answer> 0.04
16expression> asin(1-cos(0)+sin(90))
answer> 90
18expression>sin(30)^2+cos(30)^2
answer> 1
expression> end      // คำสั่งจบโปรแกรม
End program
```