204433 การแปลภาษาโปรแกรม
การบ้านที่ 1

ให้นิสิตตอบคำถามและเขียนโปรแกรมต่อไปนี้

- *โดยอนุญาติจับคู่กันทำไม่เกิน 2 คนได้*
- *ส่งไฟล์ที่เกี่ยวข้องไปที่ Google Classroom ใช้รหัสเข้า dxsfiwn โดยให้ zip ไฟล์ทั้งหมดแล้วตั้งชื่อตามรูปแบบนี้ studentID1_studentID2_HW1.zip โดย studentID1 และ studentID2 คือรหัสนิสิต*
- *การตอบคำถาม อภิปราย ให้ทำในไฟล์ README และในไฟล์นี้ควรมีการอธิบายโปรแกรมอย่างย่อด้วย*

1. จาก grammar ของ expression ทางคณิตศาสตร์ที่เราได้กล่าวถึงในห้องเรียน

   ```
   expression = ["+" | "-"] , term , {("+" | "-") , term};
   term       = factor , {("*" | "/" | "%") , factor};
   factor     = constant | "(" , expression , ")";
   constant   = digit , {digit};
   digit      = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";
   ```

   ให้นิสิตเขียน grammar ใหม่เพื่อให้รองรับ operator "^" ที่แทนการยกกำลัง และให้ว่า operator นี้มี precedence สูงที่สุด แต่ไม่สูงไปกว่าการใส่วงเล็บ

2. จากโค๊ดของ parser ที่เราได้ดูกันในชั้นเรียน ให้นิสิตใช้โค๊ดนี้เป็นหลักและเขียนเพิ่มเติมเพื่อให้โค๊ดนี้ทำการคำนวณผลลัพธ์ของ expression ทางคณิตศาสตร์ได้ ตั้งชื่อไฟล์สำหรับโค๊ดใหม่นี้ว่า expreval.c ให้ส่งไฟล์นี้พร้อมกับ test cases อีก 8 cases โดย 4 cases เป็น cases ที่คำนวณได้ถูกต้อง และ อีก 4 cases เป็น cases ที่โครงสร้างหรือ token ของ expression ไม่ถูกต้อง

3. จากโค๊ดของ parser ของ expression ทางคณิตศาสตร์ที่เราได้ดูกันในชั้นเรียน ให้นิสิตใช้โค๊ดนี้เป็นหลักและเขียนเพิ่มเติมเพื่อให้โค๊ดนี้ทำการสร้าง expression tree ตั้งชื่อโปรแกรมใหม่นี้ว่า parsetree.c ดูตัวอย่างของการนิยาม node และฟังก์ชั่นที่ใช้ในการพิมพ์ expression tree จากเลคเชอร์ เพื่อง่ายต่อการโปรแกรมในข้อนี้และข้ออื่นๆ ให้นิสิตตัดการมีเครื่องหมาย +/- หน้า term (unary plus/minus) ออก นิสิตให้เหตุผลและอธิบายได้ไหมว่า expression tree ที่สร้างขึ้น บ่งบอก precedence ของ operations ต่างๆ เช่นคือ plus minus times หรือ divide ได้อย่างไร

4. เขียนฟังก์ชั่น Prefix Infix และ Postfix สามฟังก์ชั่น ที่พิมพ์ expression ที่รับเข้ามาเริ่มต้นในรูป prefix infix และ postfix ตามลำดับ โดยให้ expression แต่ละรูปแบบทั้งสามนั้นถูกพิมพ์อยู่ในบรรทัดเดียวกัน และมีช่องว่างหนึ่งช่องว่างระหว่าง token ของ expression ในการพิมพ์รูปแบบ infix นั้น นิสิตจะต้องใส่วงเล็บตาม precedence ให้ถูกต้อง เพื่อให้การคำนวณเป็นไปตาม expression เริ่มต้นที่ใส่เข้ามา สำหรับรูป prefix และ postfix นั้น ไม่ต้องมีการใส่วงล็บใดๆ precedence ของ operation ได้ถูกบ่งไว้ภายในรูปแบบของมันอยู่แล้ว เมื่อเขียนฟังก์ชั่นทั้งสามเรียบร้อยและทดสอบจนถูกต้องแล้ว ให้เซฟโปรแกรมที่มีการเพิ่มเติมใหม่นี้ลงในไฟล์ชื่อ preinpost.c

5. ให้เพิ่มเติมโค๊ด parser (และฟังก์ชั่น Print) ของเรา ให้สามารถรับ token ที่เป็นตัวอักษรเช่น x เพื่อจะนำมาใช้เป็นตัวแปรได้ โดยอาจจะให้ kind ของ node ที่บรรจุ x นี้เป็นแบบ var จากนั้นเขียนฟังก์ชั่นเพื่อคำนวณหา derivative ของ expression เริ่มต้นเมื่อเทียบกับตัวแปร x โดย expression เริ่มต้นนี้เป็นฟังก์ชั่นของ x นิสิตไม่ควรจะทำลาย expression tree ที่ได้มาจากการ parse expression เริ่มต้น แต่ควรจะสร้าง tree ใหม่ที่เป็น tree ของ expression ที่ได้มาจากการทำ derivative เมื่อเทียบกับตัวแปร x กับ expression เริ่มต้น ให้เซฟโปรแกรมที่มีการเพิ่มเติมใหม่นี้ลงในไฟล์ชื่อ diff.c

6. จากการทำ differentiation ของ expression เราจะได้ expression ผลลัพธ์ที่เยิ่นเย้อและสามารถที่จะลดรูป (simplify) ได้พอสมควร ตัวอย่างเช่น 0 + f หรือ 0 * f โดยที่ f เป็น subtree ของ expression ให้นิสิตเขียนโค๊ดเพื่อทำการ simplify ผลลัพธ์ของ differentiation โดยใช้กฎเกณฑ์ต่ไปนี้

   0*f และ f*0 simplify เป็น 0
   0+f และ f+0 simplify เป็น f
   1*f และ f*1 simplify เป็น f
   f-f simplify เป็น 0
   f+f simplify เป็น 2*f

   ให้นิสิตรวมโค๊ดใหม่ที่เพิ่มการ simplify ผลลัพธ์เข้าไปกับโค๊ดเดิมที่เขียนเพื่อทำ differentation แล้วตั้งชื่อไฟล์ใหม่ว่า simplify.c ซึ่งจะให้เอาพุทท์เพิ่มขึ้นมาต่อไปนี้คือ

   - พิมพ์ผลลัพธ์ของการ differentiation ที่ simplify แล้วในรูป linear form
   - พิมพ์ผลลัพธ์ของการ differentiation ที่ simplify แล้วในรูป tree form

   ให้ส่งไฟล์นี้พร้อมกับ test cases อีก 8 cases โดยจะต้องให้เหตุผลด้วยว่าทำไมถึงเลือก case แต่ละcase มีความน่าสนใจอย่างไร ใช้ทดสอบอะไร เป็นกรณีทดสอบแบบ "มุมมืด" หรือไม่