**204433 การแปลภาษาโปรแกรม**
**การบ้านที่ 3**
**กำหนดส่งทาง MaxLearn ก่อนเวลา 23.50 น. ของวันอาทิตย์ที่ 8 ก.ค. 2555**

ให้นิสิตตอบคำถามและเขียนโปรแกรมต่อไปนี้

1. จากการทำ differentiation ของ expression ในการบ้านที่ 2 เราจะได้ expression ผลลัพธ์ที่เยิ่นเย้อและสามารถที่จะลดรูป (simplify) ได้พอสมควร ตัวอย่างเช่น 0 + f หรือ 0 * f โดยที่ f เป็น subtree ของ expression ให้นิสิตเขียนโค๊ดเพื่อทำการ simplify ผลลัพธ์ของ differentiation โดยใช้กฎเกณฑ์ต่ไปนี้

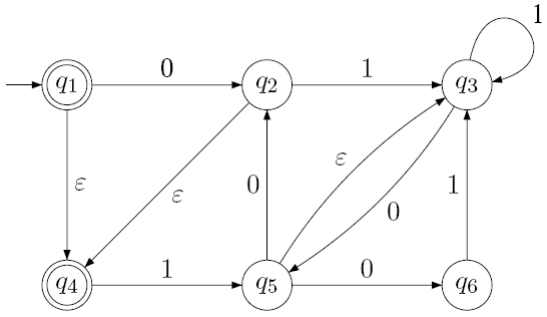
   0*f และ f*0 simplify เป็น 0
   0+f และ f+0 simplify เป็น f
   1*f และ f*1 simplify เป็น f
   f-f simplify เป็น 0
   f+f simplify เป็น 2*f

   ให้นิสิตรวมโค๊ดใหม่ที่เพิ่มการ simplify ผลลัพธ์เข้าไปกับโค๊ดเดิมที่เขียนเพื่อทำ differentation ในการบ้านก่อน แล้วตั้งชื่อไฟล์ใหม่ว่า simplify.c ซึ่งจะให้เอาพุทท์เพิ่มขึ้นมาต่อไปนี้คือ

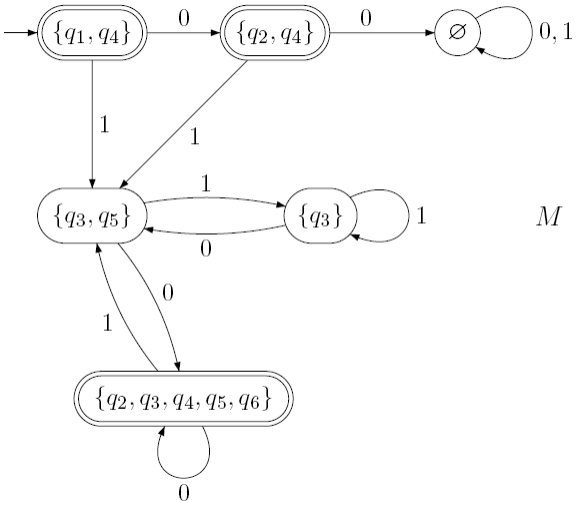
   - พิมพ์ผลลัพธ์ของการ differentiation ที่ simplify แล้วในรูป linear form
   - พิมพ์ผลลัพธ์ของการ differentiation ที่ simplify แล้วในรูป tree form

   **[ดูแนวทางคำตอบได้จากโปรแกรม simplify.c]**

2. แปลง NFA ต่อไปนี้ให้เป็น DFA



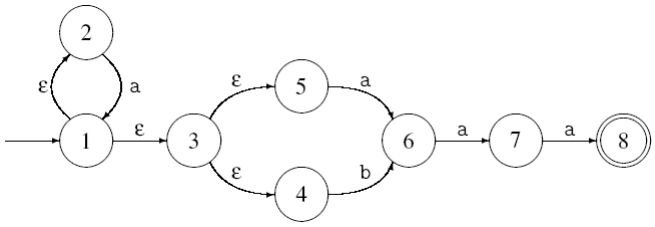

   **ได้ DFA ดังแสดงต่อไปนี้**




3. ให้ regular expression ต่อไปนี้ a*(a | b)aa จงหา

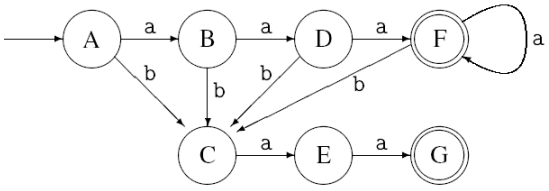
- NFA

**ได้ NFA ดังต่อไปนี้**

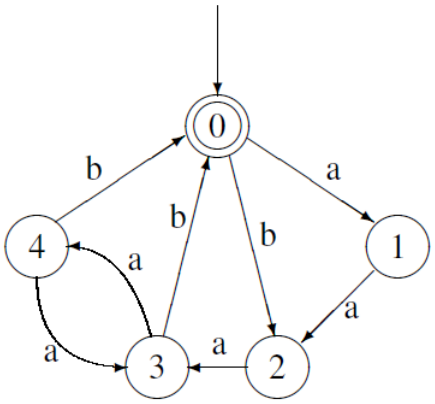



- เปลี่ยน NFA ให้เป็น DFA

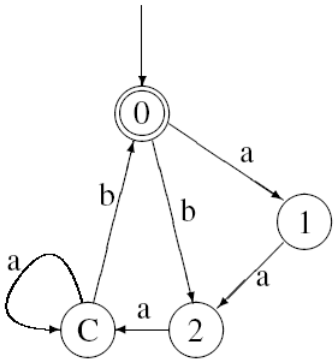
**ได้ DFA ดังต่อไปนี้**



4. ลดรูป (minimize) DFA ต่อไปนี้โดยใช้อัลกอริทึมของ Hopcroft ที่เราได้คุยกันในชั้นเรียน




**ลดรูปได้ดังต่อไปนี้**




5. ทำความเข้าใจและอธิบายการทำงานของ Maximum Munch Scanner นิสิตที่สนใจเรื่องนี้เป็นพิเศษ อาจารย์แนะนำให้อ่านบทความต่อไปนี้ที่อาจารย์ได้ให้ไว้พร้อมกับเลคเชอร์ที่ 6

   Thomas Reps, "`Maximal munch' tokenization in linear time", ACM TOPLAS, 20(2), March 1998, pp 259-273.

   ***(ดูโค๊ดในเลคเชอร์ที่ 3 ประกอบคำอธิบายต่อไปนี้ด้วย)***

   **โค๊ดที่เพิ่มเติมเข้ามาใน Maximum Munch Scanner เพื่อกำจัดการ rollback ที่มากจนเกินเหตุนั้นมีดังต่อไปนี้**

   - **เพิ่ม global counter InputPos เพื่อทำการบันทึกตำแหน่งของ input stream**
   - **เพิ่ม bit array 2 มิติ Failed เพื่อบันทึก transition ที่เป็น dead-end กล่าวคือเป็น transition ที่ไม่นำเข้าหา accepted states แถวของ Failed มีไว้สำหรับ state แต่ละ state และคอลัมน์ของ Failed มีไว้สำหรับตำแหน่งของ input stream**

เมื่อเรารู้ว่าคู่ของ <state, ตำแหน่งของ input stream> ใดที่จะนำไปสู่ dead-end แล้ว เราก็จะ break ออกจาก while loop แรกทันที การบันทึกว่าคู่ <state, ตำแหน่งของ input stream> ใด ๆจะนำไปสู่ dead-end นั้น กระทำใน while loop ที่สองขณะที่คู่ลำดับ <state, ตำแหน่งของ input stream> แต่ละอันถูก pop ออกจาก stack ไฮไลท์ในส่วนที่เพิ่มเติมต่อไปนี้

```
// recognize words
NextWord() {
    state ← s0
    lexeme ← empty string
    clear stack
    push (bad,bad)
    while (state ≠ se) do
       char ← NextChar( )
       InputPos ← InputPos + 1
       lexeme ← lexeme + char
       if Failed[state,InputPos]
            then break;
       if state ∈ SA
          then clear stack
       push (state,InputPos)
       state ← δ(state,char)
    end

    // clean up final state
    while (state ∉ SA and state ≠ bad) do
        Failed[state,InputPos) ← true
       ⟨state,InputPos⟩ ← pop()
        truncate lexeme
        roll back the input one character
    end
    // report the results
    if (state ∈ SA )
        then return lexeme
        else  return invalid
}
```