QUIZ (Time 60 minutes.)

- สร้างโฟลเดอร์ c:\temp\proglangQuiz03_(seat no.)_(ID)_(Firstname) ขึ้นมา ถ้ายังไม่ได้ทำ
    - ตัวอย่าง สำหรับคนนั่งโต๊ะหมายเลข 1: C:\temp\ proglangQuiz03_01_6510000021_Amorn
- ในโฟลเดอร์ c:\temp\proglangQuiz03_(seat no.)_(ID)_(Firstname) ใช้ IntelliJ new project ชื่อ Q1 กับ Q2 ขึ้นมา(ทำทีละโปรเจ็ค) ต้องตั้งชื่อตามนี้เป๊ะๆ ไม่งั้นตัวตรวจให้คะแนนจะไม่ตรวจให้ ต้องสร้าง scala project นะ อย่าไปสร้าง Java project
- ใน โปรเจ็ค Q1 ให้สร้าง package q1 ส่วนใน โปรเจ็ค Q2 ก็สร้าง package q2

- จากนั้น ให้เอาไฟล์โจทย์ .scala ที่โหลดใน MyCourseville copy เข้า package ของแต่ละโปรเจ็ค จะเป็นไฟล์ ชื่อ Q1.scala และ Q2.scala . Q1.scala ให้เอาใส่ โปรเจ็ค Q1 ส่วน Q2.scala ก็ให้เอาใส่โปรเจ็ค Q2
- ฟังก์ชั่นต่าง ๆ ให้เขียนแบบ Recursive เท่านั้น ห้ามใช้ลูป ถ้าไม่เขียนด้วย recursion จะได้ 0 คะแนนในข้อนั้น ๆ
- อนุญาต ให้ใช้ เมธอดของลิสต์ ได้แค่ isEmpty, length, head, tail, ::, ++ เท่านั้น ใครใช้เกินมา จะได้ 0 คะแนนในข้อนั้น ๆ
- อนุญาตให้สร้างลิสต์ โดยใช้ List(สมาชิก1,สมาชิก2,...) ได้
- เขียนเมธอดใหม่เองได้
- อนุญาตให้ access ลิสต์ด้วย index ได้
- Data structure ที่จะใช้ในโปรแกรม อนุญาตให้ใช้แต่ List เท่านั้น ถ้าใช้อย่างอื่นจะได้ 0 คะแนนในข้อที่ใช้
- เทสเคส จะรันได้คะแนนออกมาเลย แต่ตอนตรวจจะใช้เทสเคสคนละเทสกัน

1. (10 คะแนน) ทำใน Project Q1

   จงเขียนฟังก์ชั่น `fgList (f: Int=>Int, g: Int=>Int, l: List[Int]): List[Int]`

   ฟังก์ชั่นนี้ รับ `f` กับ `g` ซึ่งเป็นฟังก์ชั่น และ `l` ซึ่งเป็นลิสต์ของ `Int` ฟังก์ชั่นนี้รีเทิร์น ลิสต์ของ `f(g(x))` เมื่อ `x` คือ สมาชิกแต่ละตัว ใน `l`

   ให้พยายามใช้ `tail recursion` เพื่อลด `stack frame` (ถ้าไม่ใช้จะได้คะแนนอย่างมากแค่ครึ่งเดียว)

   ตัวอย่างเทสรัน อยู่ในไฟล์ `Q1.scala` ที่ให้แล้ว

2. (10 คะแนน)ทำในโปรเจ็ค Q2 (ข้อนี้ไม่จำเป็นต้องทำเป็น tail recursion)

   จงเขียนฟังก์ชั่นสองฟังก์ชั่น

   ฟังก์ชั่นแรกคือ

   - `replaceData(l:List[Int], pos:Int, data:Int):List[Int]`
   - ฟังก์ชั่นนี้ รับ ลิสต์ของจำนวนเต็ม l, ตำแหน่งข้อมูล pos (ให้ตำแหน่งซ้ายสุดในลิสต์ มีเลขตำแหน่งเป็น 0), และ ค่าข้อมูล data
   - ฟังก์ชั่นนี้ รีเทิร์นลิสต์ใหม่ ที่เหมือนกับ l แต่ว่า มีการ เขียน data ทับเข้าไปในตำแหน่งที่ pos
   - ให้ถือว่า ลิสต์ l ไม่มีวันเป็นลิสต์ว่าง และ pos เป็นตำแหน่งที่อยู่ในลิสต์เสมอ
   - ตัวอย่าง
     - `replaceData(List(1,2,3,4,5), 1, 9)` จะรีเทิร์น `List(1,9,3,4,5)`
     - `replaceData(List(1,2,3,4,5), 4, 9)` จะรีเทิร์น `List(1,2,3,4,9)`
     - ตัวอย่างที่เหลือ อยู่ใน เทส ของ Q2 อยู่แล้ว

   ฟังก์ชั่นที่สองคือ

   - `updateFreq(l: List[Int], freqList:List[Int]):List[Int]`
   - ฟังก์ชั่นนี้ รับ input สองตัว ตัวแรกคือลิสต์ของจำนวนเต็ม l :

QUIZ (Time 60 minutes.)

- เป็นลิสต์ที่เก็บค่าจำนวนเต็ม ตั้งแต่ 0 จนถึงค่า ที่ระบุด้วย (จำนวนสมาชิกของ freqList -1)
- ตัวอย่างเช่น ถ้า freqList มีสมาชิก 5 ตัว, ค่าที่ l จะเก็บได้ก็คือ ตั้งแต่ 0 ถึง 4 เช่น l อาจเป็น List(4,1,3,0,2,2,4)

- Input ที่สอง, freqList, คือ ลิสต์ที่เก็บ ความถี่ของแต่ละข้อมูล (ที่เก็บสะสมมาในระบบ จนถึงตอนนี้ ไม่นับรวมข้อมูลจากลิสต์ l):
  - ตัวอย่างเช่น ถ้า freqList เป็น List(3,1,0,5,3) หมายความว่า ตอนนี้มี
    - ข้อมูลเลข 0 จำนวน 3 ข้อมูล
    - ข้อมูลเลข 1 จำนวน 3 ข้อมูล
    - ข้อมูลเลข 2 จำนวน 0 ข้อมูล
    - ข้อมูลเลข 3 จำนวน 5 ข้อมูล
    - ข้อมูลเลข 4 จำนวน 3 ข้อมูล
- ฟังก์ชั่นนี้ ริเทิร์นลิสต์ใหม่ ที่เกิดจากการเอา freqList มาใส่ข้อมูลจาก ลิสต์ l เข้าไป
- ตัวอย่างเช่น ถ้า l เป็น List(4,3,2,3,4,0,3)
  freqList เป็น List(0,1,0,0,2)
- ฟังก์ชั่นนี้จะได้คำตอบเป็น ลิสต์ของความถี่ List(1,1,1,3,4)
  - 0 แต่เดิมมี 0 ตัว เพิ่มมา 1 ตัวจากลิสต์ l ในคำตอบจึงมี 1 ตัว
  - 1 แต่เดิมมี 1 ตัว เพิ่มมา 0 ตัวจากลิสต์ l ในคำตอบจึงมี 1 ตัว
  - 2 แต่เดิมมี 0 ตัว เพิ่มมา 1 ตัวจากลิสต์ l ในคำตอบจึงมี 1 ตัว
  - 3 แต่เดิมมี 0 ตัว เพิ่มมา 3 ตัวจากลิสต์ l ในคำตอบจึงมี 3 ตัว
  - 4 แต่เดิมมี 2 ตัว เพิ่มมา 2 ตัวจากลิสต์ l ในคำตอบจึงมี 4 ตัว

- เทสเคสอยู่ใน Q2.scala แล้ว