Part B (Time 30 minutes.)

- 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
- ไม่อนุญาตให้ access ลิสต์ด้วย index
- เขียนเมธอดใหม่เองจากเมธอดพื้นฐานที่อนุญาตข้างต้นได้
- ให้แยกหนึ่งข้อต่อหนึ่งไฟล์ ตั้งชื่อไฟล์ตามข้อ เช่น Question01.scala
- ในแต่ละข้อให้เขียน main เพื่อเทสได้ตามใจ อาจารย์จะตรวจโดยใช้ main ของอาจารย์เอง (มี main ตัวอย่างให้ใช้ได้)
- การส่ง ส่ง โดยเอาไฟล์ของทั้งสองข้อ zip ส่งทาง MyCourseville ของ Part B นี้ ตั้งชื่อโดยใช้ ID_PartB.zip

1. (5 คะแนน) จงเขียนฟังก์ชั่น `def swapPair(l:List[Int]): List[Int] = {`

ฟังก์ชั่นนี้ รับ `l` ซึ่งเป็นลิสต์ของ `integer` เข้ามา

รีเทิร์นลิสต์ ที่เกิดจากการสลับสมาชิกสองตัวที่อยู่ติดกัน (ถ้าไม่มีคู่ให้สลับก็ไม่ต้องเปลี่ยนอะไร)

- ถ้าไม่ใช้ `tail recursion` จะได้อย่างมาก 3.5 คะแนนเท่านั้น

ตัวอย่าง เมธอด `main` ถ้าโค้ดถูกต้อง จะได้ `output` เป็น `true` ทุกบรรทัด

```
def main(args: Array[String]): Unit = {
 val list1 = List()
 val list2 = List(22)
 val list3 = List(1,2,3)
 val list4 = List(1,2,3,4)
 val list5 = List(1,2,3,4,5)
 val list6 = List(1,2,3,4,5,6)

 println(swapPair(list1) == List())
 println(swapPair(list2) == List(22))
 println(swapPair(list3) == List(2,1,3))
 println(swapPair(list4) == List(2,1,4,3))
 println(swapPair(list5) == List(2,1,4,3,5))
 println(swapPair(list6) == List(2,1,4,3,6,5))

}
```

2. (5 คะแนน) จงเขียนฟังก์ชั่น `def partialMap(l:List[String])(f1:String => String)(f2: String => Boolean): List[String] = {`

ฟังก์ชั่นนี้ รับ พารามิเตอร์ สามชุด (ไม่ใช่พารามิเตอร์สามตัวนะ สามชุด)

- ลิสต์ของสตริง
- ฟังก์ชั่น `f1` ที่เปลี่ยนสตริง เป็นอีกสตริงหนึ่ง
- ฟังก์ชั่น `f2` ที่รับสตริงแล้วรีเทิร์น `true` หรือ `false`

สิ่งที่ฟังก์ชั่นนี้ทำ คือ รีเทิร์น ลิสต์ของสตริง ที่เกิดจากการ ใช้ ฟังก์ชั่น `f1` บนสตริงในลิสต์แต่ละตัวที่ฟังก์ชั่น `f2` เป็นจริง (ถ้าสตริงตัวไหนที่ ใช้ `f2` แล้วได้ `false` สตริงตัวนั้นจะไม่ถูกเปลี่ยน)

- ถ้าไม่ใช้ `tail recursion` จะได้อย่างมาก 3.5 คะแนนเท่านั้น

ตัวอย่างโค้ดใน `main` เป็นดังข้างล่างนี้ (ยาวหน่อยนะ) ถ้าโค้ดถูกต้องจะรันออกมาได้ `true` ทั้งหมด

```
def main(args: Array[String]): Unit = {
  val list1 = List()
  val list2 = List("2a")
  val list3 = List("1","baby","2","3","shark")
```

Part B (Time 30 minutes.)

```
  val list4 = List("anya","yor","loid","franky")

  def fChange1(x: String):String ={
    return x + "yy"
  }

  def fChange2(x:String):String ={
    return x + "punch"
  }

  def cond1(s: String):Boolean ={
    return s.length == 1
  }

  def cond2(s: String):Boolean ={
    return s.length >= 4
  }

  def cond3(s: String):Boolean ={
    return s.contains("a")
  }

  def cond4(s: String):Boolean ={
    return !s.contains("a")
  }

  val p1 = partialMap(list1)(fChange1)(_) //empty list  and "yy"
  val p2 = partialMap(list2)(fChange1)(_) // one data and "yy"
  val p3 = partialMap(list3)(fChange2)(_)
  val p4 = partialMap(list4)(fChange2)(_)

  println(p1(cond1) == List())
  println(p2(cond1) == List("2a"))
  println(p2(cond3) == List("2ayy"))
  println(p3(cond1) == List("1punch","baby","2punch","3punch","shark"))
  println(p3(cond3) == List("1","babypunch","2","3","sharkpunch"))
  println(p4(cond2) == List("anyapunch","yor","loidpunch","frankypunch"))
  println(p4(cond4) == List("anya","yorpunch","loidpunch","franky"))

}
```