**Task 1 : Subtraction**



sub = λmn.(m PRED n)

* **λ** เป็นตัวอักษรกรีกที่ใช้แทนการสร้างฟังก์ชันแบบ lambda
* **λmn.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter สองตัวคือ m และ n
* **Λmn** จะถือว่าเป็น head ส่วนข้างหลัง . จะถือว่าเป็น body
* **(m PRED n)** เป็นการใช้ฟังก์ชัน **PRED** เพื่อลดค่าของตัวแปร m และ n จนกว่าค่าของ n จะเป็น 0 แล้วคืนค่า m
* **PRED** หรือ **Predecessor** จะทำการลบค่าทีละ 1

**Scala Code:**

* ฟังก์ชัน sub รับพารามิเตอร์ m และ n ที่เป็นตัวเลขชนิด Int และมี return type เป็น Int
* ฟังก์ชัน isZero รับพารามิเตอร์ 1 ตัวเป็น Int และ return type เป็น Boolean
* เช็คเงื่อนไขว่า n เป็น 0 หรือไม่ ( isZero(n) ) ถ้า n เป็น 0 ฟังก์ชันจะคืนค่า m
* ถ้า n ไม่เป็น 0 ฟังก์ชันจะเรียกตัวเองใหม่ ( sub ) โดยลดค่า m และ n ลง 1

**Task 2 :**

**Successor: 0.25 point**



SUCC = λn.n+1

* **λn.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter ตัวเดียวคือ n
* ข้างหลัง . คือ body n+1 คือการเพิ่มค่า n 1 ค่า

**Predecessor: 0.25 point**



PRED = λn.n-1

* **λn.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter ตัวเดียวคือ n
* ข้างหลัง . คือ body n-1 คือการลดค่า n 1 ค่า

**Addition : 0.5 point**



PLUS = λmn.(m SUCC n)

* **λmn.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter สองตัวคือ m และ n
* ขั้นตอนแรกจะทำการเช็คว่าค่า n เป็น 0 หรือไม่โดยการเรียกใช้ฟังก์ isZero ถ้าเป็น 0 จะทำการคืนค่า m
* ถ้า n ไม่เป็น 0 จะทำการเรียกตัวเองโดยใส่ parameter เป็น succ(m) เพื่อเพิ่มค่า m และ pred(n) เพื่อลดค่า n

อย่างละ 1 ค่า

**Subtraction : 0.5 point**



SUB = λmn.(m PRED n)

* **λmn.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter สองตัวคือ m และ n
* ขั้นตอนแรกจะทำการเช็คว่าค่า n เป็น 0 หรือไม่โดยการเรียกใช้ฟังก์ isZero ถ้าเป็น 0 จะทำการคืนค่า m
* ถ้า n ไม่เป็น 0 จะทำการเรียกตัวเองโดยใส่ parameter เป็น pred(m) เพื่อลดค่า m และ pred(n) เพื่อลดค่า n

อย่างละ 1 ค่า

**AND : 0.5 point**



AND = λx.λy.(x y FALSE)

* **λxy.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter สองตัวคือ x และ y
* (x y FALSE) หมายถึง ถ้า x เป็นจริง ให้คืนค่า y เพราะ Boolean AND ต้องเป็นจริงทั้ง 2 ค่าจึงจะเป็นจริง การคืนค่าเลยเป็น y เพราะ y จะเป็นคำตอบให้ว่าเป็นจริงหรือเท็จ
* ถ้า x เป็นเท็จให้ใช้ค่า FALSE เพราะ Boolean AND ถ้ามีเท็จอันเดียวก็ถือว่าเป็นเท็จแล้ว

**OR : 0.5 point**



OR = λx.λy.(x TRUE y)

* **λxy.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter สองตัวคือ x และ y
* (x TRUE y) หมายถึง ถ้า x เป็นจริง ให้คืนค่า TRUE เพราะ Boolean OR แค่มีจริงสักอันก็เป็นจริงแล้ว
* ถ้า x เป็นเท็จให้คืนค่า y เพราะ จะนำค่า y มาเป็นคำตอบ

**NOT : 0.25 point**



NOT = λx.(x FALSE TRUE)

* **λx.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter ตัวเดียวคือ x
* (x FALSE TRUE) หมายถึง ถ้า x เป็นจริง ให้คืนค่า FALSE เพราะ NOT หมายถึงไม่เป็น
* ถ้า x เป็นเท็จจึงคืนค่า TRUE

**Less than or Equal : 1 point**



LEQ = λmn.isZero(sub mn)

* **λmn.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter สองตัวคือ m และ n
* ตรง else ที่เรียกใช้ตัวเองโดยใส่ parameter (pred(m), pred(n)) ที่แทนการ sub เพื่อลดค่าทีละ 1 เรื่อยๆ เพื่อนำค่าที่ลดเรื่อยๆไปเช็คในฟังก์ชัน isZero
* ทำการเช็คค่า m ก่อน n เพราะถ้าค่า m < n จริงค่า m จะเป็น 0 ก่อน n นั่นหมายความว่า m นั้นน้อยกว่า n จึงคืนค่า

TRUE

* ถ้า m ยังไม่เป็น 0 จะทำการเช็คค่า n ต่อว่าเป็น 0 หรือไม่ ถ้าเป็น 0 จริงจะคืนค่า FALSE
* ที่ไม่เช็ค Equal เพราะใช้การเช็ค m ก่อนแล้ว ยังไงถ้าเช็ค m แล้วเจอ 0 เลย ก็แสดงว่าค่า m นั้นน้อยกว่าหรือเท่ากับ n

อย่างแน่นอน

**Equality : 1 point**



EQ = λmn.(AND(LEQ mn)(LEQ nm))

* **λmn.** หมายถึงการกำหนดฟังก์ชัน lambda ที่มี parameter สองตัวคือ m และ n
* สังเกตเครื่องหมาย () เพื่อให้รู้ว่าควรทำวงเล็บไหนก่อน
* การที่ใช้ (m,n) อันแรกสลับกับ (n,m) อันสอง เพื่อให้ confirm ว่าค่าทั้งสองนั้นเท่ากันจริง
* โดยการใช้ฟังก์ชัน AND ที่จะเช็ค Boolean ทั้ง 2 มาเป็นการกำหนดค่าที่จะคืน