Brief of Assignment 1

Academic Year	2/2567
Assignment title	Lambda Calculus Operation
Assessor(s)	Dr. Issarapong Khuankrue and/or Teaching Assistant(s)
Issue Date	December 24, 2024
Submission Date	January 3, 2025 ก่อน 23.59
Score	5 Points
Course Learning Outcomes	CLO3 : สามารถอธิบาย และเขียนโปรแกรมตามหลักการในการคำนวณทาง คณิตศาสตร์แบบ Lambda Calculus
Student ID/ Name	

Assignment Brief and Guidance:

Submission Format:

- ตั้งชอเป็นี่ student_id.docx
- ส

Task 1: เขียนให้อยู่ในรูป lambda syntax สำหรับใดๆ ที่เลือกขึนมาเอง พร้อมเขียนั Scala code พร้อมคำอธิบาย(1คะแนน)

	>80%	60 – 40 %	20 - 40%	< 20%
Do the code on Lambda Calculus	สามารถอธิบาย และ เขียนโปรแกรมตามหลักการในการคำนวณทางคณิตศาสตร์แบบ Lambda Calculus ได้ อย่างชดเจน และั ถูกต้อง	สามารถอธิบายและเขียนโปรแกรม ตามหลักการในการคำนวณทางคณิตศาสตร์แบบ Lambda Calculus ได้ แต่ยังต้องการคำแนะนำเพื่อ	สามารถอธิบายและเขียนโปรแกรม ตามหลักการในการคำนวณทางคณิตศาสตร์แบบ Lambda Calculus ได้ บางสวน ต้องการ่	ไม่สามารถอธิบายและเขียนโปรแกรมได้

		ปรับปรุงเล็กน้อย	คำแนะนำเพื่อการปรับปรุงแก ้ไขจำนวนมาก	
ได้รับคะแนน	1	0.4-0.9	0.1-0.3	0

Task 2: เลือกทำฟังก์ชนั่ ใดๆตามตารางข้างล่าง พร้อมเขียนScala code พร้อมคำอธิบาย(4คะแนน)

Issarapong@it.kmitl.ac.th

Class: Functional Programming

- **Choose 1 or more functions**

- **Describe in functional abstraction (lambda syntax)**
- **Create a program in Scala**
- **Submit .docx or word file only!**

Level	Weight	Lists
Basic	**0.25**	• **Successor** • **Predecessor** • **NOT**
Medium	**0.5**	• **Addition** • **Subtraction** • **Multiplication** • **AND** • **OR** • **IsEven** • **IsOdd** • **Square** • **IsZero**

Advance	**1**	• **Exponentiation** • **Less Than or Equal to (LEQ)** • **Equality**

ทำตามเงื่อนไข weight น้ำหนักแต่ ละข้ อ	รวมกันได้ ไม่เกิน 4 คะแนน	ทำได้ 3 คะแนน	ทำได้ 2 คะแนน	ทำได้ 1 คะแนน	น้อยกว่า 1 คะแนน
ได้รับคะแนน	4	3	2	1	0

Issarapong@it.kmitl.ac.th

66070286 ปนัสยา บุญประกอบ

Task 1

Lambda equation

$$\lambda width\ \lambda length\ \lambda height.\ (width \times length \times height)$$

Scala Code

```scala
def volume(width : Double, length : Double, height : Double): Double = {
  width * length * height
}
```

volume() เป็นฟังก์ชันที่รับค่า parameter width, length, height เป็นตัวแปรชนิด Double จากนั้นคำนวณปริมาตรด้วยการคูณ width, length, height ที่รับจาก parameter เข้าด้วยกัน สุดท้ายฟังก์ชันจะ return ชนิด Double ออกมา

Task 2

Successor (0.25 คะแนน)

```scala
def successor(n: Int): Int = n + 1
```

Successor จะเพิ่มค่าของตัวเลขนั้นขึ้น 1 หน่วย ฟังก์ชันในภาษา scala จึงเป็นการบวกค่า n ขึ้น 1 เมื่อเรียกใช้ แล้วสุดท้ายจะ return ออกมาเป็น Int

Predecessor (0.25 คะแนน)

```scala
def predecessor(n: Int): Int = if (n > 0) n - 1 else 0
```

Predecessor จะลดค่าของตัวเลขที่กำหนด 1 หน่วย ฟังก์ชันในภาษา scala จึงเป็นการลดค่า n ลง 1 เมื่อเรียกใช้ แล้วสุดท้ายจะ return ออกมาเป็น Int

Addition (0.5 คะแนน)

```
def addition(m: Int, n: Int): Int = {
  if (n == 0) m
  else addition(successor(m), predecessor(n))
}
```

- ฟังก์ชัน addition จะทำงานโดยการเพิ่มค่าของ `m` ทีละ 1 ผ่าน `successor` และลดค่าของ `n` ทีละ 1 ผ่าน `predecessor` จนกว่า `n` จะถึง 0
- เมื่อ `n` เท่ากับ 0 ฟังก์ชันจะ return `m` ซึ่งจะเป็นผลลัพธ์ของ `m+n` ออกมา

Subtraction (0.5 คะแนน)

```
def subtraction(m: Int, n: Int): Int = {
  if (n == 0) m
  else subtraction(predecessor(m), predecessor(n))
}
```

- ฟังก์ชัน subtraction จะทำการลดค่า `m` และ `n` ทีละ 1 ผ่านการใช้ `predecessor` และทำการเรียกฟังก์ชันซ้ำจนกว่า `n` จะเท่ากับ 0
- เมื่อ `n` เป็น 0 ฟังก์ชันจะ return `m` ซึ่งจะเป็นผลลัพธ์ของการลบ `m-n` ออกมา

Multiplication (0.5 คะแนน)

```
def multiplication(m: Int, n: Int): Int = {
  if (n == 0) 0
  else addition(m, multiplication(m, predecessor(n)))
}
```

ฟังก์ชัน multiplication จะคูณสองจำนวน `m` และ `n` โดยการเพิ่มค่า `m` ซ้ำ ๆ จำนวน `n` ครั้ง

- เมื่อ `n == 0` ฟังก์ชันจะ return 0 (การคูณกับ 0)

- เมื่อ `n > 0` ฟังก์ชันจะเรียก `multiplication` ซ้ำ (การทำ recursion) โดยการลดค่า `n` ทีละ 1 และเพิ่มค่า `m` ทีละ 1 ด้วยฟังก์ชัน `addition`

AND (0.5 คะแนน)

```
def and(x: Boolean, y: Boolean): Boolean = {
  if (x) y else false
}
```

- ถ้า `x` เป็น `true` ฟังก์ชันจะ return `y` ที่จะเป็นได้ทั้ง `true` หรือ `false` ขึ้นอยู่กับค่าของ `y`
- ถ้า `x` เป็น `false` ฟังก์ชันจะ return false ไม่ว่า y จะเป็นอะไร

OR (0.5 คะแนน)

```
def or(x: Boolean, y: Boolean): Boolean = {
  if (x) true else y
}
```

- ถ้า `x` เป็น `true` ฟังก์ชันจะ return `true` เพราะถ้าอย่างน้อยหนึ่งค่าคือ `true` ทั้งหมดจะต้องเป็น `true`
- ถ้า `x` เป็น `false` ฟังก์ชันจะ return `y` ซึ่งจะเป็น `true` หรือ `false` ขึ้นอยู่กับ boolean ของ `y`
  - ถ้า `y` เป็น `true` ผลลัพธ์จะเป็น `true`
  - ถ้า `y` เป็น `false` ผลลัพธ์จะเป็น `false`

IsEven (0.5 คะแนน)

```
def IsEven(x: Int): Boolean = x % 2 == 0
```

- ถ้า `x` หารด้วย 2 แล้วไม่มีเศษ ฟังก์ชันจะ return `true` ซึ่งแสดงว่าเป็นเลขคู่
- ถ้า `x` หารด้วย 2 แล้วมีเศษ ฟังก์ชันจะ return `false` ซึ่งแสดงว่าเป็นเลขคี่

IsOdd (0.5 คะแนน)

```
def IsOdd(x: Int): Boolean = !IsEven(x)
```

- ถ้า `IsEven(x)` return `true` แสดงว่า `x` เป็นเลขคู่
  - ฟังก์ชัน `IsOdd` จะ return `false` เพราะเป็นเลขคู่ (ตรงข้ามกับเลขคี่)
- ถ้า `IsEven(x)` return `false` แสดงว่า `x` เป็นเลขคี่
  - ฟังก์ชัน `IsOdd` จะ return `true` เพราะเป็นเลขคี่ (ตรงข้ามกับเลขคู่)