Lab 01: hello-world

	Student ID	Name
1	66070286	ปนัสยา บุญประกอบ

Objectives:

1. (CLO1) สามารถอธิบายกระบวนคิด และหลักการในการทำงานของโปรแกรมเชิงฟังก์ชั่นทางคณิตศาสตร์

Tools:

- IDE: VS Code or IntelliJ
- JVM
- Scala 3
- Command Prompt

Exercise 1: Setup

1.1. VS Code ด้วย cs setup (recommended)
https://get-coursier.io/docs/cli-installation

1.2. IntelliJ (ถ้าเป็น computer ส่วนตัว ต้องสมัครสมาชิก แนะนำให้สมัครแบบ student)
https://www.jetbrains.com/help/idea/get-started-with-scala.html

Exercise 2: Create hello-world project

สำหรับ window:

- เปิด Command Prompt หรือ CMD
  - ตรวจสอบ `C:\Users\it>scala -version` ว่าในเครื่องมีแล้วหรือยัง
  - สร้าง folder สำหรับวางไฟล์งาน ด้วยคำสั่ง *cd ชื่อfolder*
  - พิมพ์ ว่า *sbt new scala/scala3.g8* เพื่อสร้างโปรเจคจาก template
  - ตั้งชื่อว่า *hello-world* (warning : อาจจต้องรอนาน)
  - พิมพ์ ว่า *cd hello-world*
  - พิมพ์ ว่า *sbt run*

วางภาพผลลัพธ์ที่นี่

```
C:\Users\LAB203_xx\Desktop\66070286\hello-world>sbt run
[info] [launcher] getting org.scala-sbt sbt 1.10.6  (this may take some time)...
[info] welcome to sbt 1.10.6 (Oracle Corporation Java 21.0.5)
[info] loading project definition from C:\Users\LAB203_xx\Desktop\66070286\hello-world\project
[info] loading settings for project root from build.sbt...
[info] set current project to hello-world (in build file:/C:/Users/LAB203_xx/Desktop/66070286/hello-world/)
[info] compiling 1 Scala source to C:\Users\LAB203_xx\Desktop\66070286\hello-world\target\scala-3.5.2\classes ...
[info] running hello
Hello world!
I was compiled by Scala 3. :)
[success] Total time: 3 s, completed 4 ธ.ค. 2567 13:26:23
```

- **เปิด** folder ใน IDE ที่ต้องการ แล้วแก้ไข ให้เป็น **"Hello, World!" to "Hello, ชื่อของนักศึกษา!"**
- พิมพ์ ว่า *sbt run*

วางภาพผลลัพธ์ที่นี่

```
C:\Users\LAB203_xx\Desktop\66070286\hello-world>sbt run
[info] welcome to sbt 1.10.6 (Oracle Corporation Java 21.0.5)
[info] loading project definition from C:\Users\LAB203_xx\Desktop\66070286\hello-world\project
[info] loading settings for project root from build.sbt...
[info] set current project to hello-world (in build file:/C:/Users/LAB203_xx/Desktop/66070286/hello-world/)
[info] compiling 1 Scala source to C:\Users\LAB203_xx\Desktop\66070286\hello-world\target\scala-3.5.2\classes ...
[info] running hello
Hello Noeysod
I was compiled by Scala 3. :)
[success] Total time: 3 s, completed 4 ธ.ค. 2567 13:29:32
```

Exercise 3: Uppercase Converter

ทดสอบดังต่อไปนี้

```
@main def run(): Unit =
  class Upper1:
    def convert(strings: Seq[String]): Seq[String] =
    strings.map((s: String) => s.toUpperCase)

  val up = new Upper1()
  val uppers = up.convert(List("Hello", "World!"))
  println(uppers)
```

ไม่จำเป็นต้องสร้าง project หรือ workspace ใหม่

อธิบายโปรแกรมข้างต้นทีละบรรทัด

No.	Title	Descriptions
1	class Upper1:	ประกาศสร้าง Class ชื่อ Upper1
2	def convert(strings: Seq[String]): Seq[String] =	ฟังก์ชั่นชื่อ convert รับ parameter เป็น string array
3	strings.map((s: String) => s.toUpperCase)	ใช้คำสั่ง map เพื่อเข้าถึง string แปลงตัวอักษรให้เป็น uppercase ทั้งหมด
4	val up = new Upper1()	ประกาศ attribute ใหม่ ชื่อ up เป็นตัวแปร class Upper1
5	val uppers = up.convert(List("Hello", "World!"))	ประกาศ attribute ใหม่ ชื่อ uppers โดยแปลง string จากตัวแปร up ให้เป็น List

6	println(uppers)	แสดงผลผ่าน terminal

Exercise 4: Easy Side Effect.

ทดสอบดังต่อไปนี้ (ไม่จำเป็นต้องแก้ไขcode)

```
@main def sideEffectExample(): Unit =
  var counter = 0
  def increment(): Unit =
    counter += 1

  println(s"Initial counter: $counter")
  increment()
  println(s"Counter after increment: $counter")
```

อีกโปรแกรม

```
@main def functionalExample(): Unit =
  def increment(counter: Int): Int =
    counter + 1

  val initialCounter = 0
  val updatedCounter = increment(initialCounter)

  println(s"Initial counter: $initialCounter")
  println(s"Counter after increment: $updatedCounter")
```

อภิปราย เปรียบเทียบ ลักษณะการเขียนของทั้งสองข้างต้นที่นี่

โปรแกรมแรกมีการเปลี่ยนแปลงค่า ทำให้เกิด side effects (side effects คือ การทำงานของฟังก์ชันก่อให้เกิดการเปลี่ยนแปลงภายนอกฟังก์ชัน) ไม่เหมือนโปรแกรมที่ 2 ที่มีการใช้ functional programming ซึ่งจะไม่มีการเกิด side effects และไม่มีการเปลี่ยนค่าตัวแปร (immutability)

เหตุผลก็คือการเขียนแบบ functional programming ใช้วิธีอัพเดทข้อมูลในตัวแปรใหม่แทน (ในที่นี้หมายถึงตัวแปรชื่อ updatedCounter) ไม่ได้เปลี่ยนข้อมูลในตัวแปรเดิม (ตัวแปรเดิมในที่นี้หมายถึง initialCounter) จึงมีคุณสมบัติแบบ immutable

ตัวอย่างเช่น โปรแกรมนับเลข

เมื่อเราจะเขียนโปรแกรมนับจำนวนไก่ 1 ตัว และจำนวนไข่ 1 ฟอง

sideEffectExample() จะนับไก่เป็น 1 ตัวและนับไข่ต่อไปอีกเป็น 2 เพราะใช้การเปลี่ยนค่าทับตัวแปรเก่า

แต่ functionalExample() จะนับไก่เป็น 1 และนับไข่ 1 เพราะใช้ตัวแปรใหม่ทุกครั้งที่ค่าเปลี่ยน

```
C:\Users\LAB203_xx\Desktop\66070286\hello-world>sbt run
[info] welcome to sbt 1.10.6 (Oracle Corporation Java 21.0.5)
[info] loading project definition from C:\Users\LAB203_xx\Desktop\66070286\hello-world\project
[info] loading settings for project root from build.sbt...
[info] set current project to hello-world (in build file:/C:/Users/LAB203_xx/Desktop/66070286/hello-world/)
[info] compiling 1 Scala source to C:\Users\LAB203_xx\Desktop\66070286\hello-world\target\scala-3.5.2\classes ...
[info] running sideEffectExample
Initial counter: 0
Counter after increment: 2
[success] Total time: 3 s, completed 4 ธ.ค. 2567 14:21:58
```

```
C:\Users\LAB203_xx\Desktop\66070286\hello-world>sbt run
[info] welcome to sbt 1.10.6 (Oracle Corporation Java 21.0.5)
[info] loading project definition from C:\Users\LAB203_xx\Desktop\66070286\hello-world\project
[info] loading settings for project root from build.sbt...
[info] set current project to hello-world (in build file:/C:/Users/LAB203_xx/Desktop/66070286/hello-world/)
[info] compiling 1 Scala source to C:\Users\LAB203_xx\Desktop\66070286\hello-world\target\scala-3.5.2\classes ...
[info] running functionalExample
Initial counter: 0
Counter after increment (updatedCounter1): 1
Counter after increment (updatedCounter2): 1
[success] Total time: 3 s, completed 4 ธ.ค. 2567 14:22:49
```

Exercise 5: Buy a cup of coffee (optional)

ทดสอบดังต่อไปนี้ (ไม่จำเป็นต้องแก้ไขcode)

```
class Cafe:
  def buyCoffee(cc: CreditCard): Coffee =
    val cup = Coffee()
    cc.charge(cup.price)
    cup

class CreditCard:
  def charge(price: Double): Unit =
    println("charging " + price)

class Coffee:
  val price: Double = 2.0

@main def hello(): Unit =
  val cc = CreditCard()
  val cafe = Cafe()
  val cup = cafe.buyCoffee(cc)
```

อีกโปรแกรม (ไม่จำเป็นต้องแก้ไขcode)

```
class Cafe:
  def buyCoffee(cc: CreditCard): (Coffee, Charge) ={
    val cup = new Coffee()
    (cup, Charge(cc, cup.price))
  }
case class Charge(cc: CreditCard, amount: Double):
  def combine(other: Charge): Charge =
    if cc == other.cc then
      Charge(cc, amount + other.amount)
    else
      throw Exception("Can't combine charges with different cards")

class CreditCard:
  def charge(price: Double): Unit =
    println("charging " + price)

class Coffee:
  val price: Double = 2.0

@main def hello(): Unit =
  val cc = CreditCard()
  val cafe = Cafe()
  val cup = cafe.buyCoffee(cc)
```

อภิปราย เปรียบเทียบ ลักษณะการเขียนของทั้งสองข้างต้นที่นี่

คำอธิบาย