Session 10 Lab Report

Title: Product Analyzer using Kotlin Classes and Higher-Order Functions

Objective:

To design and implement a Product Analyzer system using Kotlin data classes, collections, higher-order functions, and lambda expressions to apply discounts and filter products dynamically.

Concepts Covered:

- Data Classes in Kotlin
- List Collections
- Higher-Order Functions
- Lambda Expressions
- map() and filter() operations
- Immutability using copy()

Complete Implementation Code:

```
data class Product(
    val name: String,
    val category: String,
    val price: Double,
    val stock: Int
)

fun applyDiscount(
    products: List<Product>,
    discountFunction: (Double) -> Double
): List<Product> {
    return products.map { product ->
        product.copy(price = discountFunction(product.price))
    }
}

fun filterProducts(
    products: List<Product>,
    condition: (Product) -> Boolean
): List<Product> {
    return products.filter { condition(it) }
}

fun main() {
    val products = listOf(
```

```
        Product("Laptop", "Electronics", 100000.0, 5),
        Product("Phone", "Electronics", 50000.0, 15),
        Product("Chair", "Furniture", 2000.0, 25),
        Product("Table", "Furniture", 3000.0, 5),
        Product("Pen", "Stationery", 50.0, 100)
    )

    val discountedProducts = applyDiscount(products) { price ->
        price * 0.9
    }

    println("--- Discounted Product List ---")
    discountedProducts.forEach {
        println("Name: ${it.name} | Category: ${it.category} | Price: ${it.price} | Stock: ${it.stock}")
    }

    val lowStockProducts = filterProducts(products) {
        it.stock < 10
    }

    println("\n--- Low Stock Products (<10) ---")
    lowStockProducts.forEach {
        println("${it.name} (${it.stock})")
    }

    val premiumProducts = filterProducts(products) {
        it.price > 1000
    }

    println("\n--- Premium Products (>1000) ---")
    premiumProducts.forEach {
        println("${it.name} (${it.price})")
    }
}
```

Learning Outcome:

After completing this lab, students gained practical knowledge of combining object-oriented and functional programming concepts in Kotlin. They learned how to build reusable logic using higher-order functions and lambda expressions while maintaining clean and modular code structure.