**Lab 5- Inline Functions in Kotlin**

Below is a lab exercise on inline functions in Kotlin. This exercise focuses on understanding the basics of inline functions and their usage.

// Lab Exercise: Inline Functions in Kotlin

// Task 1: Create an inline function named 'executeOperation' that takes two integers

// and a lambda function as parameters. It should apply the lambda to the numbers and return the result.

inline fun executeOperation(a: Int, b: Int, operation: (Int, Int) -> Int): Int {

return operation(a, b)

}

// Task 2: Declare a lambda function named 'add' that takes two integers as parameters

// and returns their sum.

val add: (Int, Int) -> Int = { x, y -> x + y }

// Task 3: Declare a lambda function named 'multiply' that takes two integers as parameters

// and returns their product.

val multiply: (Int, Int) -> Int = { x, y -> x \* y }

fun main() {

// Task 4: Call the 'executeOperation' function with different numbers and the 'add' lambda.

val resultAdd = executeOperation(10, 5, add)

println("Result of addition: $resultAdd")

// Task 5: Call the 'executeOperation' function with different numbers and the 'multiply' lambda.

val resultMultiply = executeOperation(4, 6, multiply)

println("Result of multiplication: $resultMultiply")

// Task 6: Declare an inline function named 'logExecution' that takes a function as a parameter

// and logs a message before and after executing the function.

inline fun logExecution(message: String, function: () -> Unit) {

println("Executing: $message")

function()

println("Execution completed.")

}

// Task 7: Use the 'logExecution' function to wrap a simple function and observe the logs.

logExecution("Print message") {

println("Hello from the wrapped function!")

}

}

**Instructions:**

* Create the inline function 'executeOperation' as instructed in the comments.
* Declare the lambda functions 'add' and 'multiply' as instructed in the comments.
* In the main function, call the 'executeOperation' function with different numbers and the 'add' and 'multiply' lambdas.
* Create the inline function 'logExecution' as instructed in the comments.
* Use the 'logExecution' function to wrap a simple function and observe the logs.

This exercise is designed to reinforce your understanding of inline functions and their usage in Kotlin.