[2CEIT5PE18: MOBILE APPLICATION DEVELOPMENT]

Practical: 1

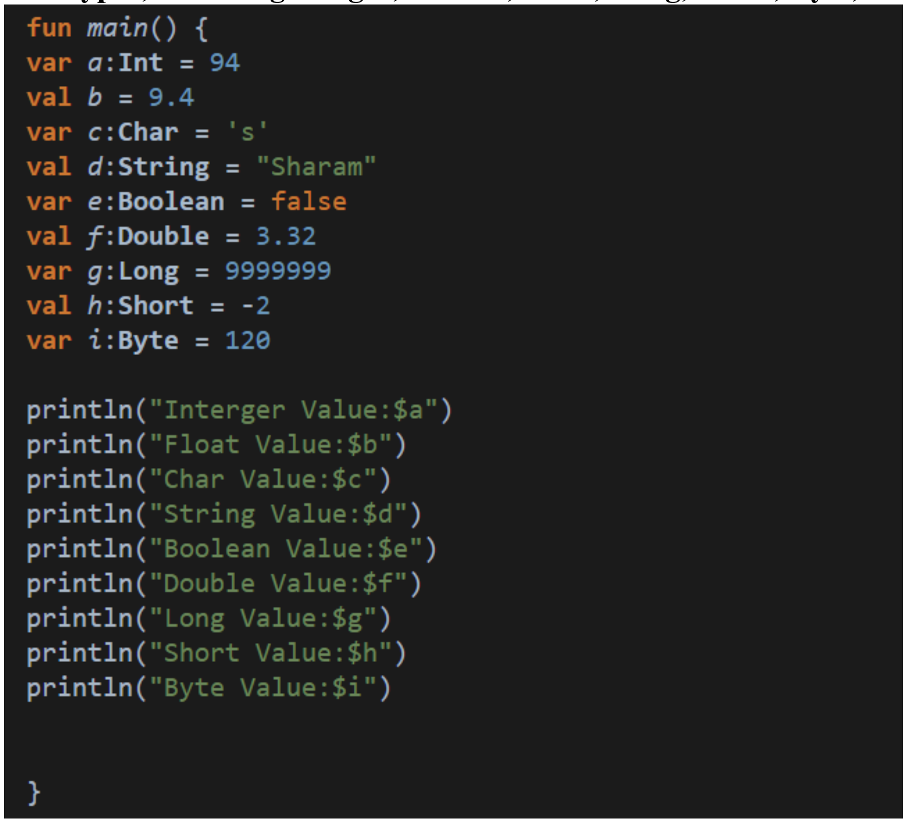
AIM:- Develop a Kotlin program for demonstrating various programming concepts.

Submitted By: Manthan Sharma
Enrollment number: 22012011104

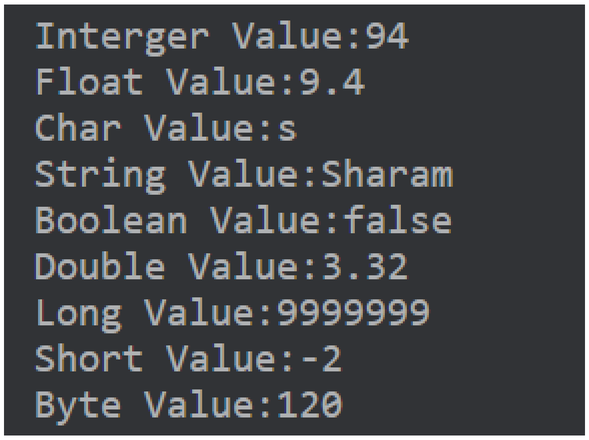
Ganpat University | U.V. Patel College of Engineering
॥ विद्यया समाजोत्कर्षः ॥

Department of Computer Engineering/Information Technology

1.1. Store & Display Values in Different Variables: Create and display variables of different data types, including Integer, Double, Float, Long, Short, Byte, Char, Boolean, and String



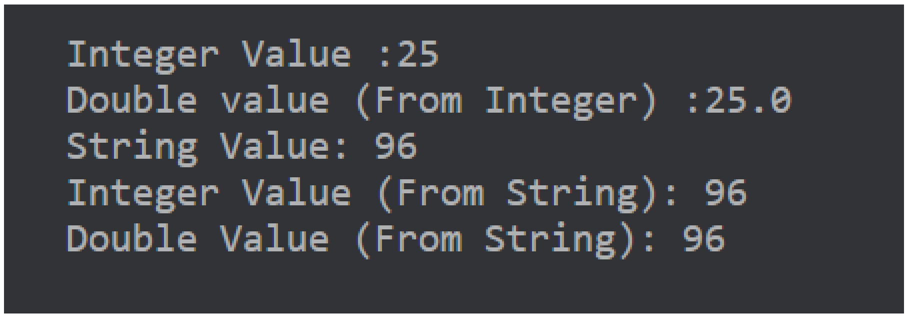
```
fun main() {
var a:Int = 94
val b = 9.4
var c:Char = 's'
val d:String = "Sharam"
var e:Boolean = false
val f:Double = 3.32
var g:Long = 9999999
val h:Short = -2
var i:Byte = 120

println("Interger Value:$a")
println("Float Value:$b")
println("Char Value:$c")
println("String Value:$d")
println("Boolean Value:$e")
println("Double Value:$f")
println("Long Value:$g")
println("Short Value:$h")
println("Byte Value:$i")


}
```



```
Interger Value:94
Float Value:9.4
Char Value:s
String Value:Sharam
Boolean Value:false
Double Value:3.32
Long Value:9999999
Short Value:-2
Byte Value:120
```

1.2. Type Conversion: Perform type conversions such as Integer to Double, String to Integer, and String to Double.

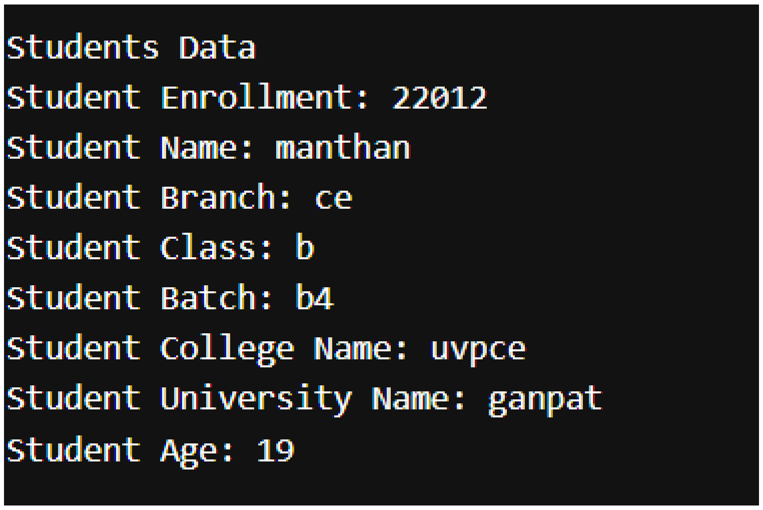
```
fun main(){
   val a = 25
   println("Integer Value :$a")
   val b = a.toDouble()
   println("Double value (From Integer) :$b")

   val c:String = "96"
   println("String Value: $c")
   val d = c.toInt()
   val e = c.toDouble()
   println("Integer Value (From String): $d")
   println("Double Value (From String): $d")
   }
```



```
Integer Value :25
Double value (From Integer) :25.0
String Value: 96
Integer Value (From String): 96
Double Value (From String): 96
```

1.3. Scan student's information and display all the data: Input and display data of students, including their name, enrolment no, branch,etc.

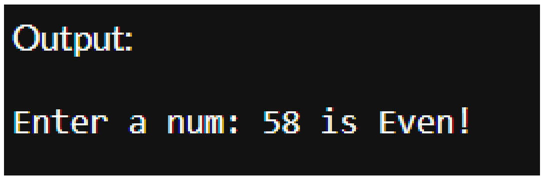
```
fun main() {
   println("Student Enrollment: ")
    val enno = readLine()
    println("Student Name: ")
    val name =  readLine()
    println("Student Branch: ")
    val branch =  readLine()
    println("Student Class: ")
    val cls =  readLine()
    println("Student Batch: ")
    val batch =  readLine()
    println("Student College Name: ")
    val Clg_name =  readLine()
    println("Student University Name: ")
    val Uni_name =  readLine()
    println("Student Age: ")
    val Age =  readLine()
    println()
    println("*****************************")
    println()
    println("Students Data")
    println("Student Enrollment: $enno")
    println("Student Name: $name")
    println("Student Branch: $branch")
    println("Student Class: $cls")
    println("Student Batch: $batch")
    println("Student College Name: $Clg_name")
    println("Student University Name: $Uni_name")
    println("Student Age: $Age")
}
```



```
Students Data
Student Enrollment: 22012
Student Name: manthan
Student Branch: ce
Student Class: b
Student Batch: b4
Student College Name: uvpce
Student University Name: ganpat
Student Age: 19
```

1.4. Check Odd or Even Numbers: Determine whether a number is odd or even using control flow within println() method.

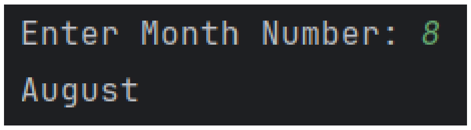
```
  fun main(){
  print("Enter a num: ")
  val x = readLine()!!.toInt()
  when(x%2){
  0 -> println("$x is Even!") 1 -> println("$x is Odd!")
  }
  if (x%2 == 0){
  "$x is Even!"
  }
  else{

  "$x is Odd!"

  }
}
```



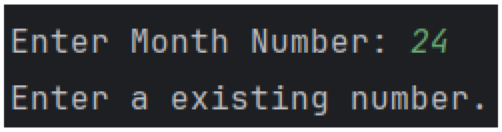
```
Output:

Enter a num: 58 is Even!
```

1.5. Display Month Name: Use a when expression to display the month name based on user input.

```
fun main(){
   print("Enter Month Number: ")
   val x = readln()!!.toInt()
   when(x){
      1 -> print("January")
      2 -> print("February")
      3 -> print("March")
      4 -> print("April")
      5 -> print("May")
      6 -> print("June")
      7 -> print("July")
      8 -> print("August")
      9 -> print("September")
      10 -> print("October")
      11 -> print("November")
      12 -> print("December")
      else -> {
         print("Enter a existing number.")
      }
   }
}
```



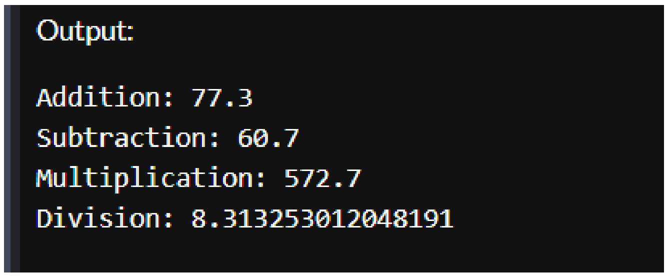
```
Enter Month Number: 8
August
```



```
Enter Month Number: 24
Enter a existing number.
```

1.6. User-Defined Function: Create a user-defined function to perform arithmetic operations (addition, subtraction, multiplication, division) on two numbers

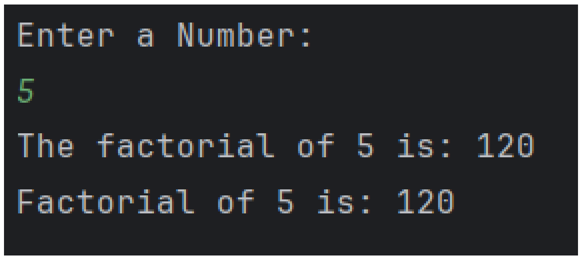
```
  fun arithmeticOperation(num1: Double, num2: Double, operation: String): Double {
return when (operation) {
"add" -> num1 + num2
"subtract" -> num1 - num2
"multiply" -> num1 * num2
"divide" -> {
 if (num2 != 0.0) num1 / num2
 else throw IllegalArgumentException("Division by zero is not allowed")
 }
 else -> throw IllegalArgumentException("Invalid operation")
 }
}
fun main() {
 val num1 = 69.0
 val num2 = 8.3
 println("Addition: ${arithmeticOperation(num1, num2, "add")}")
 println("Subtraction: ${arithmeticOperation(num1, num2, "subtract")}")
 println("Multiplication: ${arithmeticOperation(num1, num2, "multiply")}")
 println("Division: ${arithmeticOperation(num1, num2, "divide")}")
}
```



```
Output:

Addition: 77.3
Subtraction: 60.7
Multiplication: 572.7
Division: 8.313253012048191
```

1.7. Factorial Calculation with Recursion: Calculate the factorial of a number using recursion.

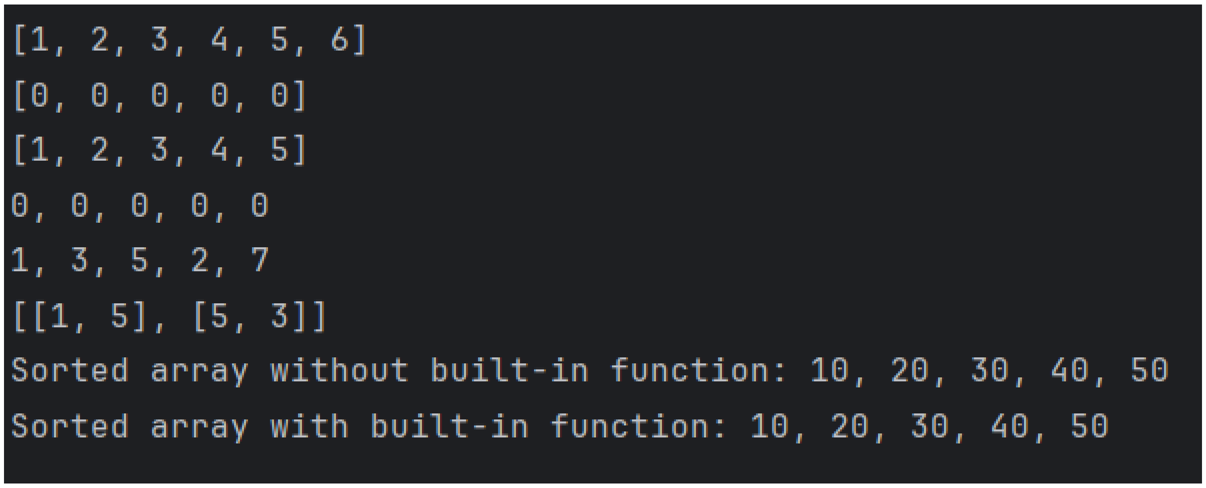
```
fun main(){
    println("Enter a Number:")
    val x = readln().toInt()
    println("The factorial of $x is: "+Fac(x))
    val number = 5
    val result = factorial(number)
    println("Factorial of $number is: $result")
}
fun Fac(a:Int):Int{
    if (a<=1){
        return 1
    }
    else{
        return a*Fac(a-1)
    }
}
tailrec fun factorial(n: Int, result: Long = 1): Long {
    return if (n == 1) {
        result
    } else {
        factorial(n - 1, result * n)
    }
}
```



```
Enter a Number:
5
The factorial of 5 is: 120
Factorial of 5 is: 120
```

1.8. Working with Arrays: Explore array operations such as Arrays.deepToString(), contentDeepToString(), IntArray.joinToString(), and use them to print arrays. Utilize variousloop types like range, downTo, until, etc., to manipulate arrays. Sort an array of integers bothwithout using built-in functions and with built-in functions.

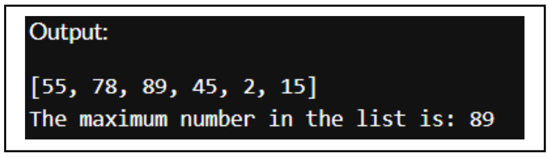
```
package Practicals.src

fun main(){
    var ar1 = arrayOf(1,2,3,4,5,6)
    println(ar1.contentDeepToString())
    var ar2 = Array<Int>(5){0}
    println(ar2.contentDeepToString())
    val ar3 = Array<Int>(5){inx -> inx+1}
    println(ar3.contentDeepToString())
    var ar4 = IntArray(5){0}
    println(ar4.joinToString())
    var ar5 = intArrayOf(1,3,5,2,7)
    println(ar5.joinToString())
    var ar6 = arrayOf(intArrayOf(1,5), intArrayOf(5,3))
    println(ar6.contentDeepToString())
    val intArray = intArrayOf(50, 30, 40, 10, 20)
    bubbleSort(intArray)
    println("Sorted array without built-in function: ${intArray.joinToString()}")
    intArray.sort()
    println("Sorted array with built-in function: ${intArray.joinToString()}")
}
fun bubbleSort(arr: IntArray) {
    val n = arr.size
    for (i in 0 until n - 1) {
        for (j in 0 until n - i - 1) {
            if (arr[j] > arr[j + 1]) {
                // Swap arr[j] and arr[j+1]
                val temp = arr[j]
                arr[j] = arr[j + 1]
                arr[j + 1] = temp
            }
        }
    }
}
```



```
[1, 2, 3, 4, 5, 6]
[0, 0, 0, 0, 0]
[1, 2, 3, 4, 5]
0, 0, 0, 0, 0
1, 3, 5, 2, 7
[[1, 5], [5, 3]]
Sorted array without built-in function: 10, 20, 30, 40, 50
Sorted array with built-in function: 10, 20, 30, 40, 50
```

1.9. Find Maximum Number from ArrayList: Write a program to find the maximum number from an ArrayList of integers.

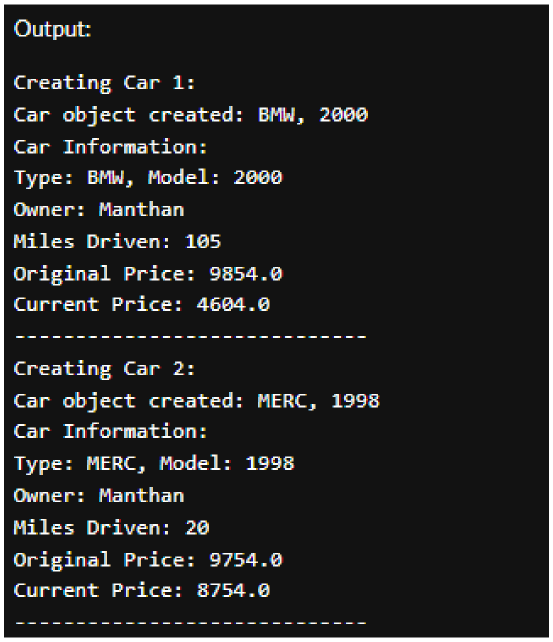
```
fun main() {

val numbers = arrayListOf(55,78,89,45,2,15)
println(numbers)

if (numbers.isEmpty()) {
    println("The list is empty.")
    return
  }

  var maxNumber = numbers[0]


  for (num in numbers) {
      if (num > maxNumber) {
          maxNumber = num
      }
  }

  println("The maximum number in the list is: $maxNumber")
}
```



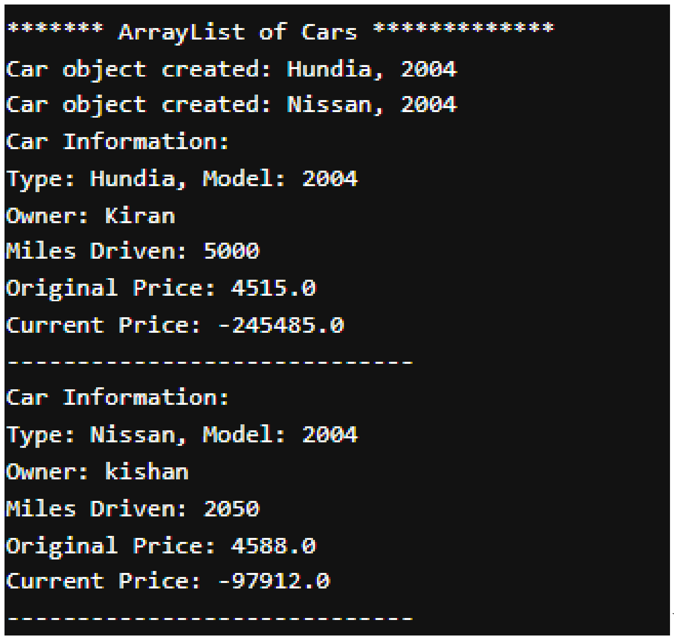
```
Output:

[55, 78, 89, 45, 2, 15]
The maximum number in the list is: 89
```

1.10. Class and Constructor Creation: Define different classes and constructors. Create a "Car" class with properties like type, model, price, owner, and miles driven. Implement functions to get car information, original car price, current car price, and display car information

```
 class Car(
    private val type: String,
    private val model: Int,
    private val originalPrice: Double,
    private val owner: String,
    private val milesDriven: Int
) {
    // Init block for logging
    init {
        println("Car object created: $type, $model")
    }

    // Calculate the current price based on miles driven
    fun getCurrentPrice(): Double = originalPrice -
(milesDriven * 50.0)

    // Display the car information
    fun getCarInformation() {
        println("""
            Car Information:
            Type: $type, Model: $model
            Owner: $owner
            Miles Driven: $milesDriven
            Original Price: $originalPrice
            Current Price: ${getCurrentPrice()}
            ----------------------------
        """.trimIndent())
    }
}

fun main() {
    // Creating and displaying individual car objects
    println("Creating Car 1:")
    val car1 = Car("BMW", 2000, 9854.0, "Manthan",
105)
    car1.getCarInformation()

    println("Creating Car 2:")
    val car2 = Car("MERC", 1998, 9754.0,
"Manthan", 20)
    car2.getCarInformation()

    // Creating and displaying cars from an ArrayList
    println("******* ArrayList of Cars
*************")
    val carList = arrayListOf(
        Car("Hundia", 2004, 4515.0, "Kiran", 5000),
        Car("Nissan", 2004, 4588.0, "kishan", 2050)
    )

    carList.forEach { it.getCarInformation() }
}
```



```
Output:

Creating Car 1:
Car object created: BMW, 2000
Car Information:
Type: BMW, Model: 2000
Owner: Manthan
Miles Driven: 105
Original Price: 9854.0
Current Price: 4604.0
-----------------------------
Creating Car 2:
Car object created: MERC, 1998
Car Information:
Type: MERC, Model: 1998
Owner: Manthan
Miles Driven: 20
Original Price: 9754.0
Current Price: 8754.0
-----------------------------
```



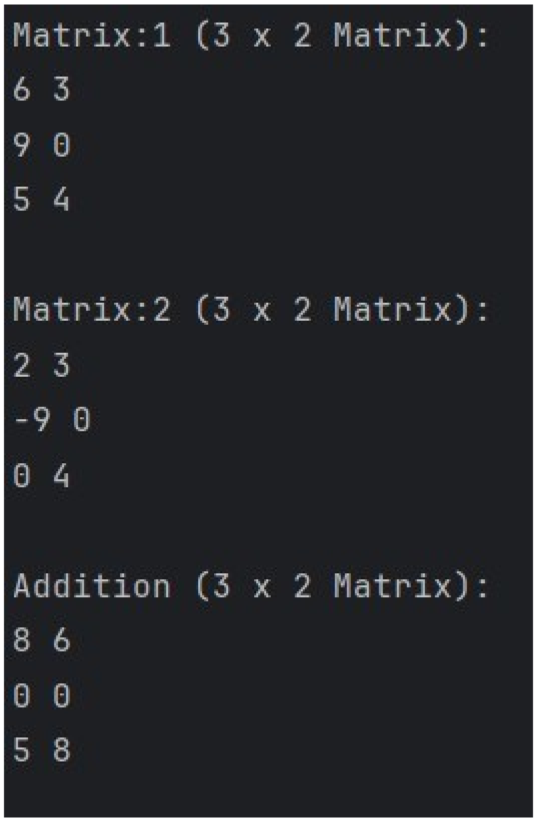
```
******* ArrayList of Cars *************
Car object created: Hundia, 2004
Car object created: Nissan, 2004
Car Information:
Type: Hundia, Model: 2004
Owner: Kiran
Miles Driven: 5000
Original Price: 4515.0
Current Price: -245485.0
-----------------------------
Car Information:
Type: Nissan, Model: 2004
Owner: kishan
Miles Driven: 2050
Original Price: 4588.0
Current Price: -97912.0
-----------------------------
```

}

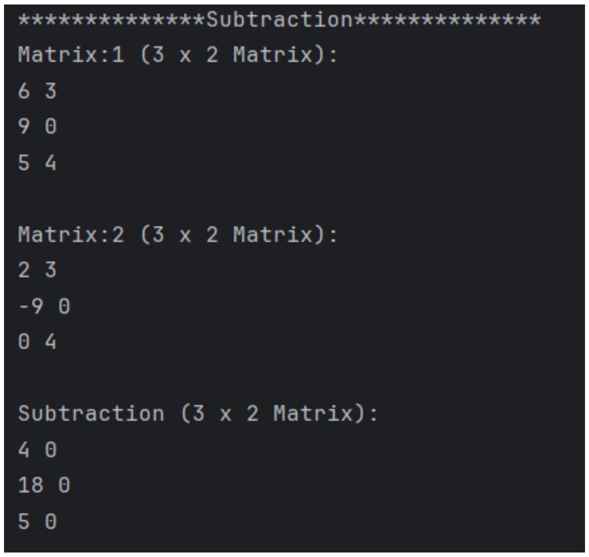
1.11. Operator Overloading and Matrix Operations: Explain operator overloading and

implement matrix addition, subtraction, and multiplication using a "Matrix" class. Overloadthe toString() function in the "Matrix" class for customized output.

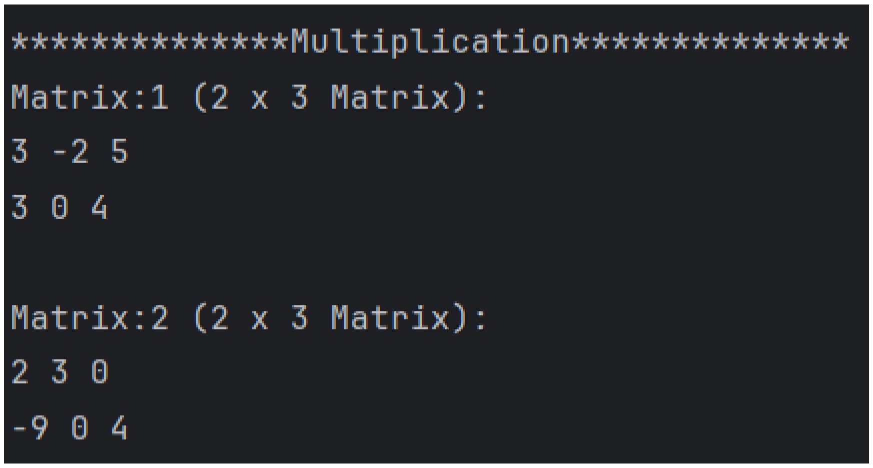
```
class Matrix(private val rows: Int, private val cols: Int, private val matrix: Array<IntArray>) {

    // Overload the plus (+) operator for matrix addition
    operator fun plus(other: Matrix): Matrix {
        val result = Array(rows) { IntArray(cols) }
        for (i in 0 until rows) {
            for (j in 0 until cols) {
                result[i][j] = this.matrix[i][j] + other.matrix[i][j]
            }
        }
        return Matrix(rows, cols, result)





    }

    // Overload the minus (-) operator for matrix subtraction
    operator fun minus(other: Matrix): Matrix {
        val result = Array(rows) { IntArray(cols) }
        for (i in 0 until rows) {
            for (j in 0 until cols) {
                result[i][j] = this.matrix[i][j] - other.matrix[i][j]
            }
        }
        return Matrix(rows, cols, result)
    }

    // Overload the times (*) operator for matrix multiplication
    operator fun times(other: Matrix): Matrix {
        val result = Array(rows) { IntArray(other.cols) }
        for (i in 0 until rows) {
            for (j in 0 until other.cols) {
                for (k in 0 until cols) {
                    result[i][j] += this.matrix[i][k] * other.matrix[k][j]
                }
            }
        }
        return Matrix(rows, other.cols, result)
    }

    // Overload the toString() function for custom output of the matrix
    override fun toString(): String {
        val builder = StringBuilder()
        for (i in 0 until rows) {
            for (j in 0 until cols) {
                builder.append("${matrix[i][j]} ")
            }
            builder.append("\n")
        }
        return builder.toString()
    }

    // Function to print matrix with dimensions
    fun printMatrix(label: String) {
        println("$label ($rows x $cols Matrix):")
        println(this.toString())
    }
}

fun main() {
    // Creating matrices for different operations



    val firstMatrix = Matrix(3, 2, arrayOf(
        intArrayOf(6, 3),
        intArrayOf(9, 0),
        intArrayOf(5, 4)
    ))

    val secondMatrix = Matrix(3, 2, arrayOf(
        intArrayOf(2, 3),
        intArrayOf(-9, 0),
        intArrayOf(0, 4)
    ))

    val thirdMatrix = Matrix(2, 3, arrayOf(
        intArrayOf(3, -2, 5),
        intArrayOf(3, 0, 4)
    ))

    val fourthMatrix = Matrix(2, 3, arrayOf(
        intArrayOf(2, 3, 0),
        intArrayOf(-9, 0, 4)
    ))

    // Addition
    println("*************Addition*************")
    firstMatrix.printMatrix("Matrix:1")
    secondMatrix.printMatrix("Matrix:2")
    val additionResult = firstMatrix + secondMatrix
    additionResult.printMatrix("Addition")

    // Subtraction
    println("*************Subtraction*************")
    firstMatrix.printMatrix("Matrix:1")
    secondMatrix.printMatrix("Matrix:2")
    val subtractionResult = firstMatrix - secondMatrix
    subtractionResult.printMatrix("Subtraction")

    // Multiplication
    println("*************Multiplication*************")
    thirdMatrix.printMatrix("Matrix:1")
    fourthMatrix.printMatrix("Matrix:2")
    val multiplicationResult = thirdMatrix * fourthMatrix
    multiplicationResult.printMatrix("Multiplication")
}
```



```
Matrix:1 (3 x 2 Matrix):
6 3
9 0
5 4

Matrix:2 (3 x 2 Matrix):
2 3
-9 0
0 4

Addition (3 x 2 Matrix):
8 6
0 0
5 8
```



```
**************Subtraction**************
Matrix:1 (3 x 2 Matrix):
6 3
9 0
5 4

Matrix:2 (3 x 2 Matrix):
2 3
-9 0
0 4

Subtraction (3 x 2 Matrix):
4 0
18 0
5 0
```



```
**************Multiplication**************
Matrix:1 (2 x 3 Matrix):
3 -2 5
3 0 4

Matrix:2 (2 x 3 Matrix):
2 3 0
-9 0 4
```