[2CEIT5PE18: MOBILE APPLICATION DEVELOPMENT]

Practical: 1

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

Submitted By: Siddivinayak Doppalapudi
Enrollment number: 23012531069

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

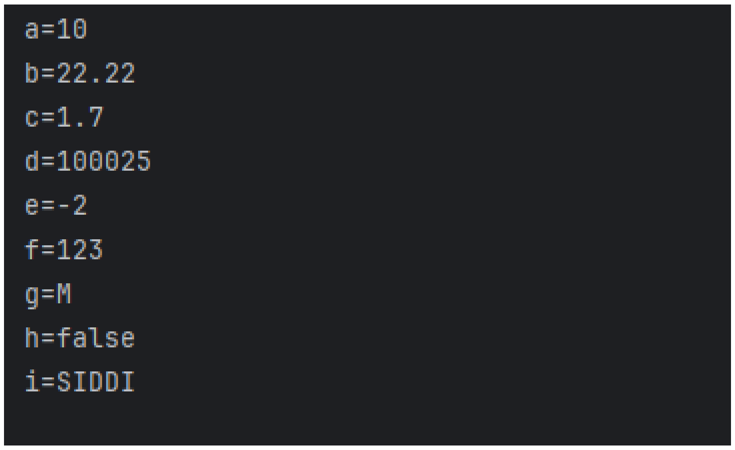
Department of Computer Engineering/Information Technology

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.**

Answer:

```
fun main() {
    var a: Int=10
    var b: Double=22.22
    var c: Float=1.7f
    var d: Long=100025
    var e: Short=-2
    var f: Byte=123
    var g: Char= 'M'
    var h: Boolean= false
    var i: String= "SIDDI"

    println("a=$a\nb=$b\nc=$c\nd=$d\ne=$e\nf=$f\ng=$g\nh=$h\ni=$i")
}
```

Output:



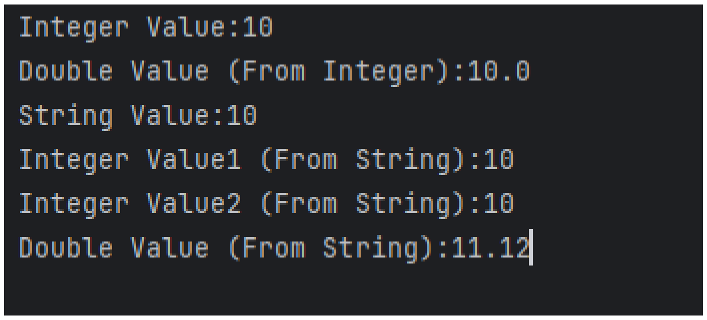
```
a=10
b=22.22
c=1.7
d=100025
e=-2
f=123
g=M
h=false
i=SIDDI
```

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

Answer:

```
fun main() {
    val intValue: Int = 10
    val doubleFromInt: Double = intValue.toDouble()
    val stringValue: String = "10"
    val intFromString1: Int = stringValue.toInt()
    val intFromString2: Int = "10".toInt()
    val doubleFromString: Double = "11.12".toDouble()

    println("Integer Value:$intValue")
    println("Double Value (From Integer):$doubleFromInt")
    println("String Value:$stringValue")
    println("Integer Value1 (From String):$intFromString1")
    println("Integer Value2 (From String):$intFromString2")
    println("Double Value (From String):$doubleFromString")
}
```

Output:



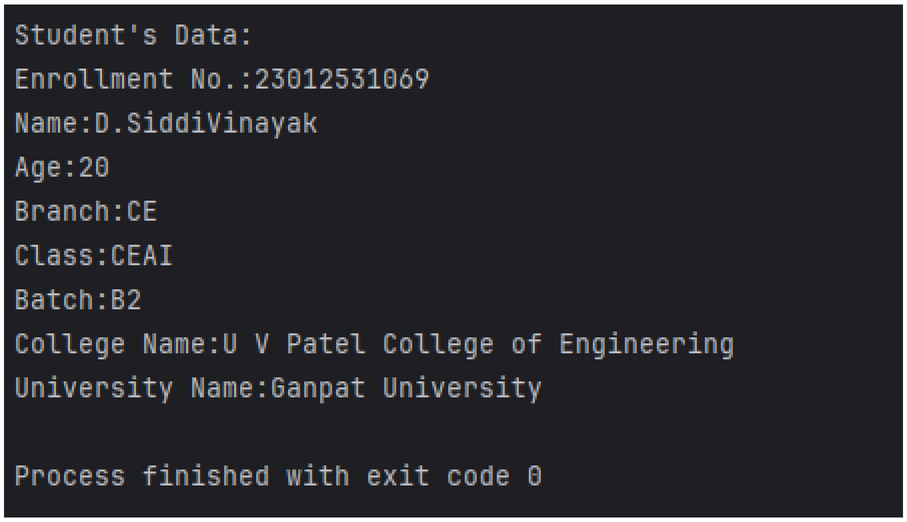
```
Integer Value:10
Double Value (From Integer):10.0
String Value:10
Integer Value1 (From String):10
Integer Value2 (From String):10
Double Value (From String):11.12
```

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

Answer:

```
fun main() {
    val studentEnrollmentNo = "23012531069"
    val studentName = "D.SiddiVinayak"
    val studentBranch = "CE"
    val studentClass = "CEAI"
    val studentBatch = "B2"
    val studentCollegeName = "U V Patel College of Engineering"
    val studentUniversityName = "Ganpat University"
    val studentAge = 20

    println("student Enrollment No.:$studentEnrollmentNo")
    println("student Name:$studentName")
    println("student Branch:$studentBranch")
    println("student Class:$studentClass")
    println("student Batch:$studentBatch")
    println("student College Name:$studentCollegeName")
    println("student University Name:$studentUniversityName")
    println("student Age:$studentAge")

    println("\n*******************\n")

    println("Student's Data:")
    println("Enrollment No.:$studentEnrollmentNo")
    println("Name:$studentName")
    println("Age:$studentAge")
    println("Branch:$studentBranch")
    println("Class:$studentClass")
    println("Batch:$studentBatch")
    println("College Name:$studentCollegeName")
    println("University Name:$studentUniversityName")
}
```

Output:



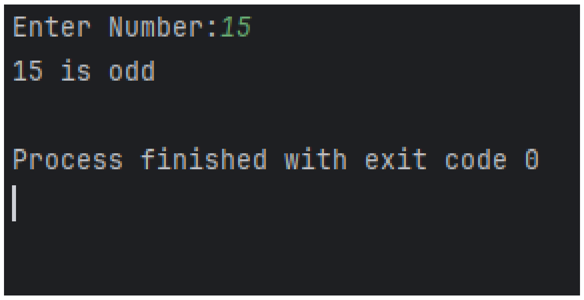
```
Student's Data:
Enrollment No.:23012531069
Name:D.SiddiVinayak
Age:20
Branch:CE
Class:CEAI
Batch:B2
College Name:U V Patel College of Engineering
University Name:Ganpat University

Process finished with exit code 0
```

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

Answer:

```
import java.util.Scanner

fun main() {
    val reader = Scanner(System.`in`)
    print("Enter Number:")

    val number = reader.nextInt()

    if (number % 2 == 0) {
        println("$number is even")
    } else {
        println("$number is odd")
    }
}
```

output:



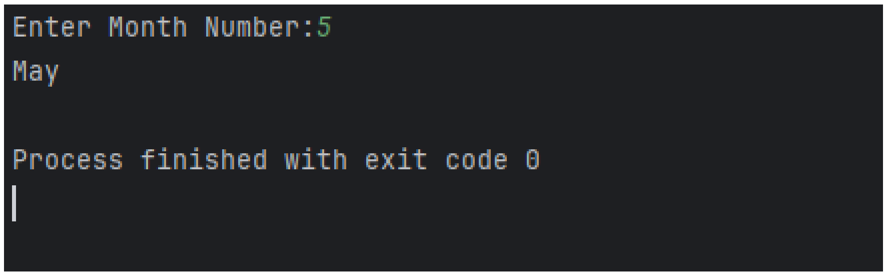
```
Enter Number:15
15 is odd

Process finished with exit code 0
```

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

Answer:

```
import java.util.Scanner

fun main() {
    val reader = Scanner(System.`in`)

    print("Enter Month Number:")
    val monthNumber = reader.nextInt()

    val monthName = when (monthNumber) {
        1 -> "January"
        2 -> "February"
        3 -> "March"
        4 -> "April"
        5 -> "May"
        6 -> "June"
        7 -> "July"
        8 -> "August"
        9 -> "September"
        10 -> "October"
        11 -> "November"
        12 -> "December"
        else -> "please enter proper number"
    }
    println(monthName)
}
```

output:



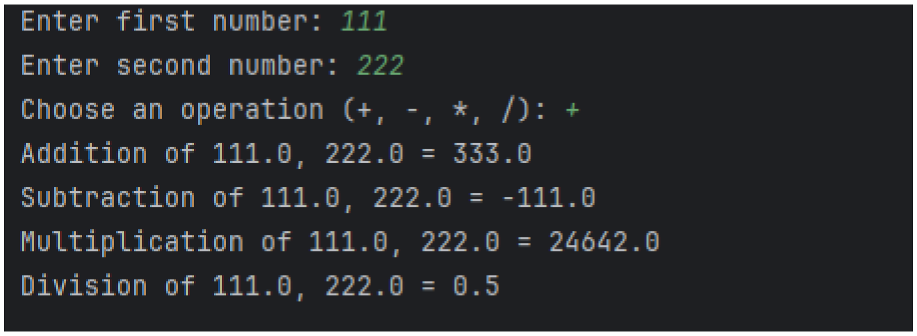
```
Enter Month Number:5
May

Process finished with exit code 0
```

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

Answer:

```
import java.util.Scanner

fun main() {
    val reader = Scanner(System.`in`)

    print("Enter first number: ")
    val num1 = reader.nextDouble()

    print("Enter second number: ")
    val num2 = reader.nextDouble()

    print("Choose an operation (+, -, *, /): ")
    val operator = reader.next()

    // Call the user-defined function to perform the calculation
    val result = performArithmetic(num1, num2, operator)

    // Display the result
    if (result != null) {
        println("Addition of $num1, $num2 = ${num1 + num2}")
        println("Subtraction of $num1, $num2 = ${num1 - num2}")
        println("Multiplication of $num1, $num2 = ${num1 * num2}")
        if (num2 != 0.0) {
            println("Division of $num1, $num2 = ${num1 / num2}")
        } else {
            println("Division by zero is not allowed.")
        }
    } else {
        println("Error: Invalid operator or division by zero.")
    }
}


fun performArithmetic(num1: Double, num2: Double, operator: String): Double? {
    return when (operator) {
        "+" -> num1 + num2
        "-" -> num1 - num2
        "*" -> num1 * num2
        "/" -> {
            if (num2 != 0.0) {
                num1 / num2
            } else {
                null // Division by zero
            }
        }
        else -> null // Invalid operator
    }
}
```

output:



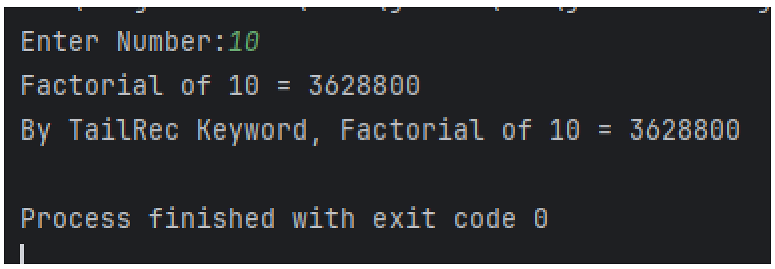
```
Enter first number: 111
Enter second number: 222
Choose an operation (+, -, *, /): +
Addition of 111.0, 222.0 = 333.0
Subtraction of 111.0, 222.0 = -111.0
Multiplication of 111.0, 222.0 = 24642.0
Division of 111.0, 222.0 = 0.5
```

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

Answer:

```
import java.util.Scanner

fun main() {
    val reader = Scanner(System.`in`)
    print("Enter Number:")

    val number = reader.nextInt()

    if (number < 0) {
        println("Factorial is not defined for negative numbers.")
    } else {
        // Calculate factorial using a regular recursive function
        val factorialResult = factorial(number)
        println("Factorial of $number = $factorialResult")

        // Calculate factorial using a tail-recursive function
        val tailrecFactorialResult = tailFactorial(number)
        println("By TailRec Keyword, Factorial of $number =
$tailrecFactorialResult")
    }
}

fun factorial(n: Int): Long {
    return if (n == 0 || n == 1) {
        1
    } else {
        n * factorial(n - 1)
    }
}

tailrec fun tailFactorial(n: Int, accumulator: Long = 1): Long {
    return if (n == 0 || n == 1) {
        accumulator
    } else {
        tailFactorial(n - 1, accumulator * n)
    }
}
```

output:



```
Enter Number:10
Factorial of 10 = 3628800
By TailRec Keyword, Factorial of 10 = 3628800

Process finished with exit code 0
```

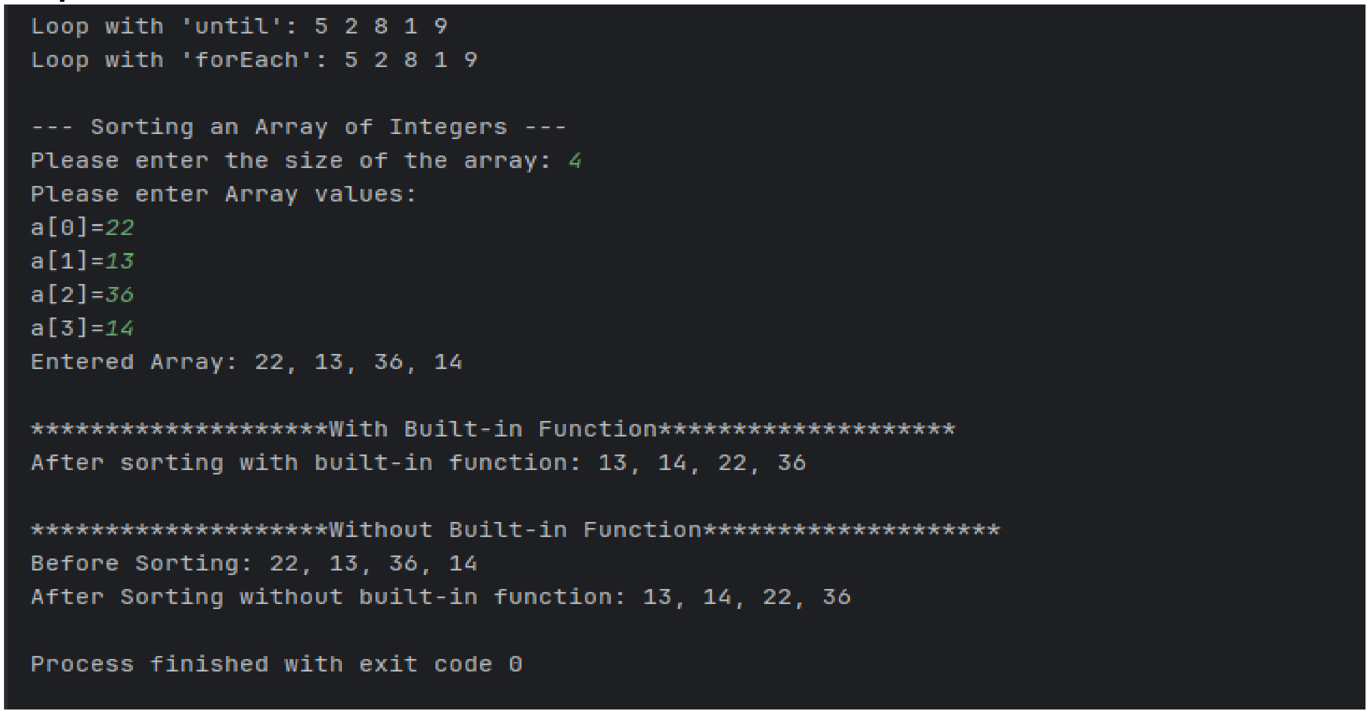
1.8. Working with Arrays: **Explore array operations such as Arrays.deepToString(), contentDeepToString(), IntArray.joinToString(), and use them to print arrays. Utilize various loop types like range, downTo, until, etc., to manipulate arrays. Sort an array of integers both without using built-in functions and with built-in functions.**

Answer:

```
import java.util.Scanner

fun main() {
    val reader = Scanner(System.`in`)

    // --- 1. Array Creation and Printing ---

    println("--- Array Creation and Printing ---")

    // Create Array-1 by using arrayOf() method:
    val array1 = arrayOf(10, 90, 60, 80, 100)
    println("Create Array-1 by using arrayOf() method:")
    println(array1.contentDeepToString()) // For 1D array, contentDeepToString()
works fine

    // Create Array-2 by using Array<>() with default values (all zeros)
    val array2 = Array<Int>(5) { 0 }
    println("\nCreate Array-2 by using Array<>():")
    println(array2.contentDeepToString())

    // Create Array-3 by using Array<>() and Lambda function (initializing with
index values)
    val array3 = Array<Int>(8) { i -> i }
    println("\nCreate Array-3 by using Array<>() and Lambda function:")
    println(array3.contentDeepToString())

    // Create Array-4 by using IntArray() (primitive int array)
    val array4 = IntArray(4) { 0 }
    println("\nCreate Array-4 by using IntArray():")
    println(array4.joinToString()) // IntArray has joinToString() directly

    // Create Array-5 by using intArrayOf() (primitive int array with specified
values)
```

```
    val array5 = intArrayOf(12, 10, 1, 5, 18, 19)
    println("\nCreate Array-5 by using intArrayOf():")
    println(array5.joinToString())

    // Create 2D Array-6 by using arrayOf() and intArrayOf()
    val array2D = arrayOf(intArrayOf(1, 3), intArrayOf(4, 5), intArrayOf(6, 7))
    println("\nCreate 2D Array-6 by using arrayOf() and intArrayOf():")
    println(array2D.contentDeepToString()) // contentDeepToString() is essential for
2D arrays


    // --- 2. Array Manipulation with Loop Types ---

    println("\n--- Array Manipulation with Loop Types ---")
    val numbers = intArrayOf(5, 2, 8, 1, 9)

    println("Original array for loops: ${numbers.joinToString()}")

    // Using 'range' loop (for i in 0..numbers.size - 1)
    print("Loop with 'range' (0..size-1): ")
    for (i in 0..numbers.size - 1) {
        print("${numbers[i]} ")
    }
    println()

    // Using 'downTo' loop (for i in numbers.size - 1 downTo 0)
    print("Loop with 'downTo': ")
    for (i in numbers.size - 1 downTo 0) {
        print("${numbers[i]} ")
    }
    println()

    // Using 'until' loop (for i in 0 until numbers.size)
    print("Loop with 'until': ")
    for (i in 0 until numbers.size) {
        print("${numbers[i]} ")
    }
    println()

    // Using 'forEach' loop (Kotlin's idiomatic way to iterate)
    print("Loop with 'forEach': ")
    numbers.forEach { print("$it ") }
    println()


    // --- 3. Sorting an Array of Integers ---

    println("\n--- Sorting an Array of Integers ---")

    // Get user input for an array
    print("Please enter the size of the array: ")
    val size = reader.nextInt()
    val userArray = IntArray(size)

    println("Please enter Array values:")
    for (i in 0 until size) {
        print("a[$i]=")
        userArray[i] = reader.nextInt()
    }

    println("Entered Array: ${userArray.joinToString()}")

    // --- Sorting with built-in function ---
    println("\n*******************With Built-in Function*******************")
    val sortedWithBuiltIn = userArray.clone() // Create a copy to sort, keeping
original for custom sort
    sortedWithBuiltIn.sort() // Sorts the array in-place
    println("After sorting with built-in function:
${sortedWithBuiltIn.joinToString()}")

    // --- Sorting without built-in function (Bubble Sort example) ---
    println("\n*******************Without Built-in Function*******************")
    val sortedWithoutBuiltIn = userArray.clone() // Create a copy of the original
array
    println("Before Sorting: ${sortedWithoutBuiltIn.joinToString()}")

    // Implement Bubble Sort
    for (i in 0 until sortedWithoutBuiltIn.size - 1) {
        for (j in 0 until sortedWithoutBuiltIn.size - i - 1) {
            if (sortedWithoutBuiltIn[j] > sortedWithoutBuiltIn[j + 1]) {
                // Swap elements
                val temp = sortedWithoutBuiltIn[j]
                sortedWithoutBuiltIn[j] = sortedWithoutBuiltIn[j + 1]
                sortedWithoutBuiltIn[j + 1] = temp
            }
        }
    }
    println("After Sorting without built-in function:
${sortedWithoutBuiltIn.joinToString()}")
}
```

output:

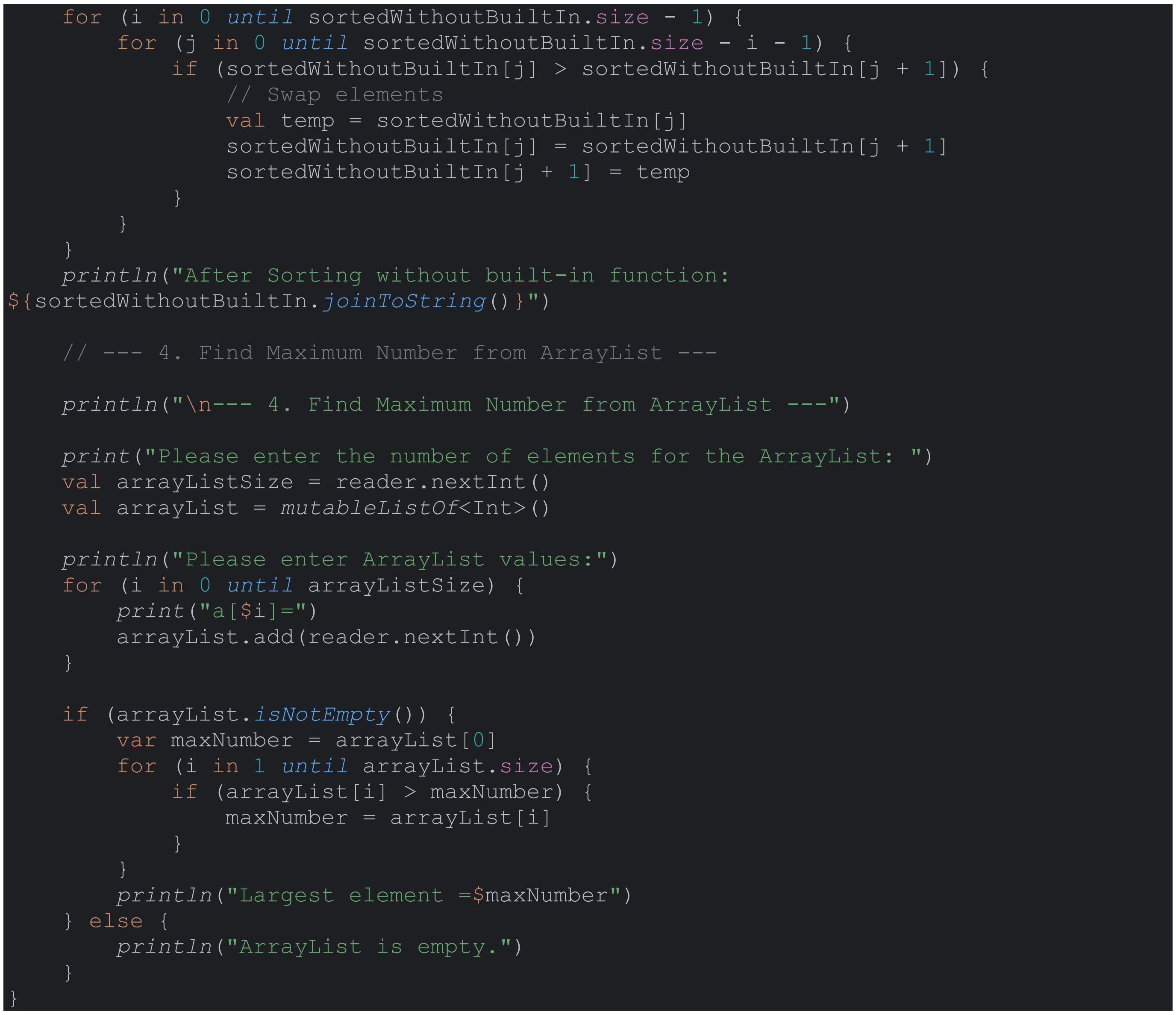


```
Loop with 'until': 5 2 8 1 9
Loop with 'forEach': 5 2 8 1 9

--- Sorting an Array of Integers ---
Please enter the size of the array: 4
Please enter Array values:
a[0]=22
a[1]=13
a[2]=36
a[3]=14
Entered Array: 22, 13, 36, 14

********************With Built-in Function********************
After sorting with built-in function: 13, 14, 22, 36

********************Without Built-in Function********************
Before Sorting: 22, 13, 36, 14
After Sorting without built-in function: 13, 14, 22, 36

Process finished with exit code 0
```

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

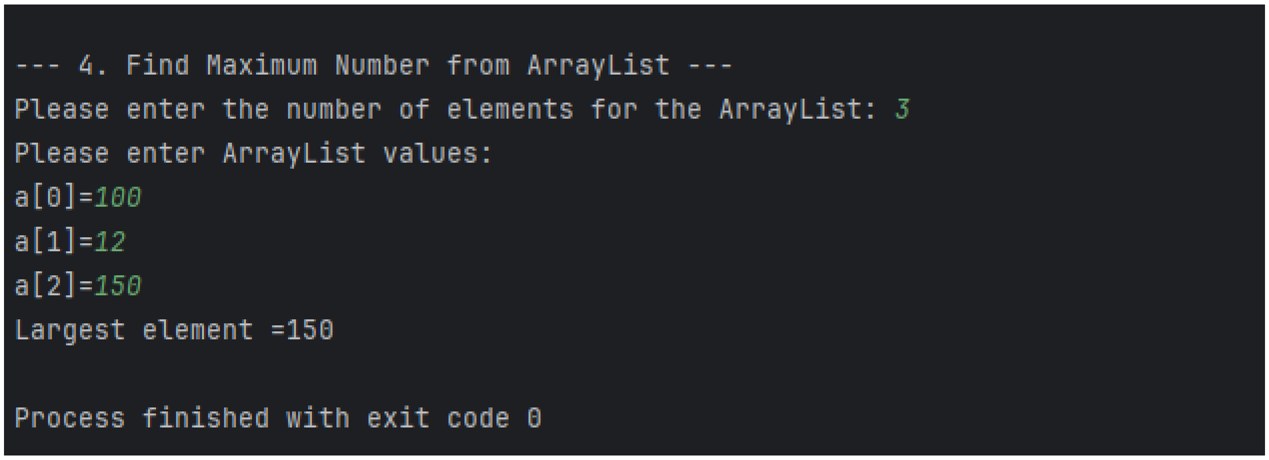
Answer:

```
import java.util.Scanner

fun main() {
    val reader = Scanner(System.`in`)

    // --- 1. Array Creation and Printing ---

    println("--- Array Creation and Printing ---")

    // Create Array-1 by using arrayOf() method:
    val array1 = arrayOf(10, 90, 60, 80, 100)
    println("Create Array-1 by using arrayOf() method:")
    println(array1.contentDeepToString()) // For 1D array, contentDeepToString()
works fine

    // Create Array-2 by using Array<>() with default values (all zeros)
    val array2 = Array<Int>(5) { 0 }
    println("\nCreate Array-2 by using Array<>():")
    println(array2.contentDeepToString())

    // Create Array-3 by using Array<>() and Lambda function (initializing with
index values)
    val array3 = Array<Int>(8) { i -> i }
    println("\nCreate Array-3 by using Array<>() and Lambda function:")
    println(array3.contentDeepToString())

    // Create Array-4 by using IntArray() (primitive int array)
    val array4 = IntArray(4) { 0 }
    println("\nCreate Array-4 by using IntArray():")
    println(array4.joinToString()) // IntArray has joinToString() directly

    // Create Array-5 by using intArrayOf() (primitive int array with specified
values)
    val array5 = intArrayOf(12, 10, 1, 5, 18, 19)
    println("\nCreate Array-5 by using intArrayOf():")
    println(array5.joinToString())

    // Create 2D Array-6 by using arrayOf() and intArrayOf()
    val array2D = arrayOf(intArrayOf(1, 3), intArrayOf(4, 5), intArrayOf(6, 7))
    println("\nCreate 2D Array-6 by using arrayOf() and intArrayOf():")
    println(array2D.contentDeepToString()) // contentDeepToString() is essential for
2D arrays


    // --- 2. Array Manipulation with Loop Types ---

    println("\n--- Array Manipulation with Loop Types ---")
    val numbers = intArrayOf(5, 2, 8, 1, 9)

    println("Original array for loops: ${numbers.joinToString()}")

    // Using 'range' loop (for i in 0..numbers.size - 1)
    print("Loop with 'range' (0..size-1): ")
    for (i in 0..numbers.size - 1) {
        print("${numbers[i]} ")
    }
    println()

    // Using 'downTo' loop (for i in numbers.size - 1 downTo 0)
    print("Loop with 'downTo': ")
    for (i in numbers.size - 1 downTo 0) {
        print("${numbers[i]} ")
    }
    println()

    // Using 'until' loop (for i in 0 until numbers.size)
    print("Loop with 'until': ")
    for (i in 0 until numbers.size) {
        print("${numbers[i]} ")
    }
    println()

    // Using 'forEach' loop (Kotlin's idiomatic way to iterate)
    print("Loop with 'forEach': ")
    numbers.forEach { print("$it ") }
    println()


    // --- 3. Sorting an Array of Integers ---

    println("\n--- Sorting an Array of Integers ---")

    // Get user input for an array
    print("Please enter the size of the array: ")
    val size = reader.nextInt()
    val userArray = IntArray(size)

    println("Please enter Array values:")
    for (i in 0 until size) {
        print("a[$i]=")
        userArray[i] = reader.nextInt()
    }

    println("Entered Array: ${userArray.joinToString()}")

    // --- Sorting with built-in function ---
    println("\n*******************With Built-in Function*******************")
    val sortedWithBuiltIn = userArray.clone() // Create a copy to sort, keeping
original for custom sort
    sortedWithBuiltIn.sort() // Sorts the array in-place
    println("After sorting with built-in function:
${sortedWithBuiltIn.joinToString()}")

    // --- Sorting without built-in function (Bubble Sort example) ---
    println("\n*******************Without Built-in Function*******************")
    val sortedWithoutBuiltIn = userArray.clone() // Create a copy of the original
array
    println("Before Sorting: ${sortedWithoutBuiltIn.joinToString()}")

    // Implement Bubble Sort
```

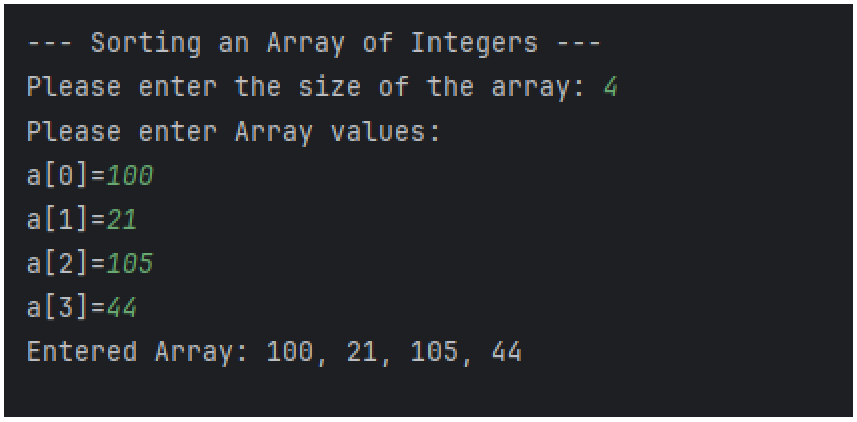


```
    for (i in 0 until sortedWithoutBuiltIn.size - 1) {
        for (j in 0 until sortedWithoutBuiltIn.size - i - 1) {
            if (sortedWithoutBuiltIn[j] > sortedWithoutBuiltIn[j + 1]) {
                // Swap elements
                val temp = sortedWithoutBuiltIn[j]
                sortedWithoutBuiltIn[j] = sortedWithoutBuiltIn[j + 1]
                sortedWithoutBuiltIn[j + 1] = temp
            }
        }
    }
    println("After Sorting without built-in function:
${sortedWithoutBuiltIn.joinToString()}")

    // --- 4. Find Maximum Number from ArrayList ---

    println("\n--- 4. Find Maximum Number from ArrayList ---")

    print("Please enter the number of elements for the ArrayList: ")
    val arrayListSize = reader.nextInt()
    val arrayList = mutableListOf<Int>()

    println("Please enter ArrayList values:")
    for (i in 0 until arrayListSize) {
        print("a[$i]=")
        arrayList.add(reader.nextInt())
    }

    if (arrayList.isNotEmpty()) {
        var maxNumber = arrayList[0]
        for (i in 1 until arrayList.size) {
            if (arrayList[i] > maxNumber) {
                maxNumber = arrayList[i]
            }
        }
        println("Largest element =$maxNumber")
    } else {
        println("ArrayList is empty.")
    }
}
```

output:



```

--- 4. Find Maximum Number from ArrayList ---
Please enter the number of elements for the ArrayList: 3
Please enter ArrayList values:
a[0]=100
a[1]=12
a[2]=150
Largest element =150

Process finished with exit code 0
```

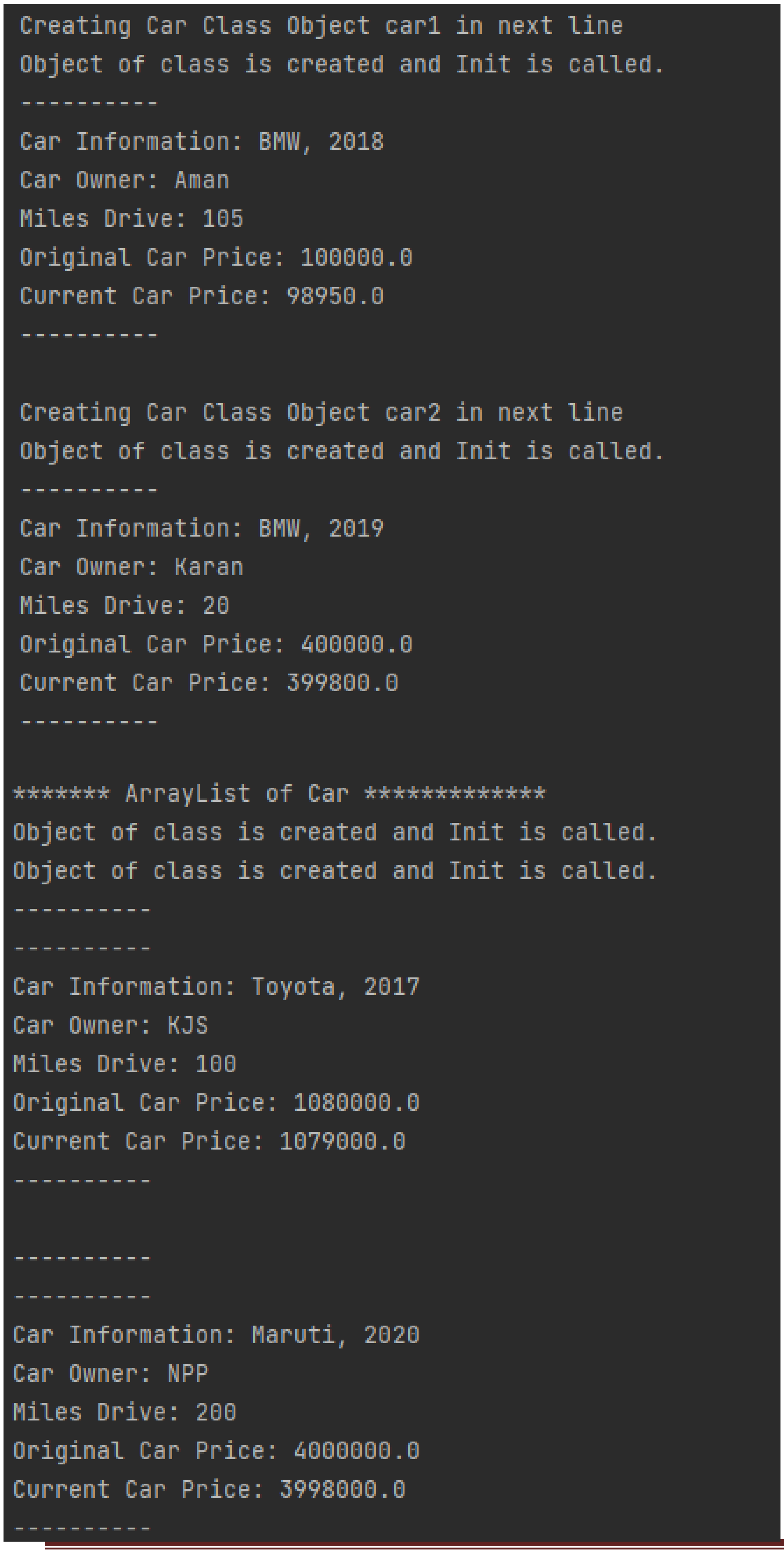

```
--- Sorting an Array of Integers ---
Please enter the size of the array: 4
Please enter Array values:
a[0]=100
a[1]=21
a[2]=105
a[3]=44
Entered Array: 100, 21, 105, 44
```

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.**

Answer:

```
import java.util.Scanner

// Define the Car class
class Car(
    val type: String,
    val model: Int,
    val price: Double,
    var owner: String,
    var milesDriven: Double
) {
    // Initializer block - called when an object of the class is created
    init {
        println("Object of class is created and Init is called.")
    }

    // Function to get car information
    fun getCarInformation(): String {
        return "$type, $model"
    }

    // Function to get the original car price
    fun getOriginalCarPrice(): Double {
        return price
    }

    // Function to calculate and get the current car price
    fun getCurrentCarPrice(): Double {
        // Simple depreciation logic: 1% depreciation per 100 miles driven
        val depreciationPerMile = price * 0.01 / 100
        return price - (milesDriven * depreciationPerMile)
    }

    // Function to display all car information
    fun displayCarInformation() {
        println("Car Information: ${getCarInformation()}")
        println("Car Owner: $owner")
        println("Miles Drive: ${milesDriven.toInt()}") // Display as integer as in
example
        println("Original Car Price: ${getOriginalCarPrice()}")
        println("Current Car Price: ${getCurrentCarPrice()}")
        println("----------")
    }
}

fun main() {
    val reader = Scanner(System.`in`)

    // --- Creating Car objects and displaying information ---

    println("Creating Car Class Object car1 in next line.")
    val car1 = Car("BMW", 2018, 100000.0, "Aman", 105.0)
    car1.displayCarInformation()

    println("Creating Car Class Object car2 in next line.")
    val car2 = Car("BMW", 2019, 400000.0, "Karan", 20.0)
    car2.displayCarInformation()

    // --- Working with an ArrayList of Car objects ---
    println("\n****** ArrayList of Car *************")

    val carList = arrayListOf<Car>()

    // Adding cars to the ArrayList
    // Object 1
    println("Object of class is created and Init is called.")
    carList.add(Car("Toyota", 2017, 1080000.0, "KJS", 100.0))

    // Object 2
    println("Object of class is created and Init is called.")
    carList.add(Car("Maruti", 2020, 400000.0, "NPP", 200.0))

    // Displaying information for cars in the ArrayList
    carList.forEach { car ->
        car.displayCarInformation()
    }

    reader.close()
}
```

output:



```
 Creating Car Class Object car1 in next line
 Object of class is created and Init is called.
 ----------
 Car Information: BMW, 2018
 Car Owner: Aman
 Miles Drive: 105
 Original Car Price: 100000.0
 Current Car Price: 98950.0
 ----------

 Creating Car Class Object car2 in next line
 Object of class is created and Init is called.
 ----------
 Car Information: BMW, 2019
 Car Owner: Karan
 Miles Drive: 20
 Original Car Price: 400000.0
 Current Car Price: 399800.0
 ----------

******* ArrayList of Car *************
Object of class is created and Init is called.
Object of class is created and Init is called.
----------
----------
Car Information: Toyota, 2017
Car Owner: KJS
Miles Drive: 100
Original Car Price: 1080000.0
Current Car Price: 1079000.0
----------


----------
----------
Car Information: Maruti, 2020
Car Owner: NPP
Miles Drive: 200
Original Car Price: 4000000.0
Current Car Price: 3998000.0
----------
```

1.11. Operator Overloading and Matrix Operations: **Explain operator overloading and implement matrix addition, subtraction, and multiplication using a "Matrix" class. Overload the toString() function in the "Matrix" class for customized**

Output:

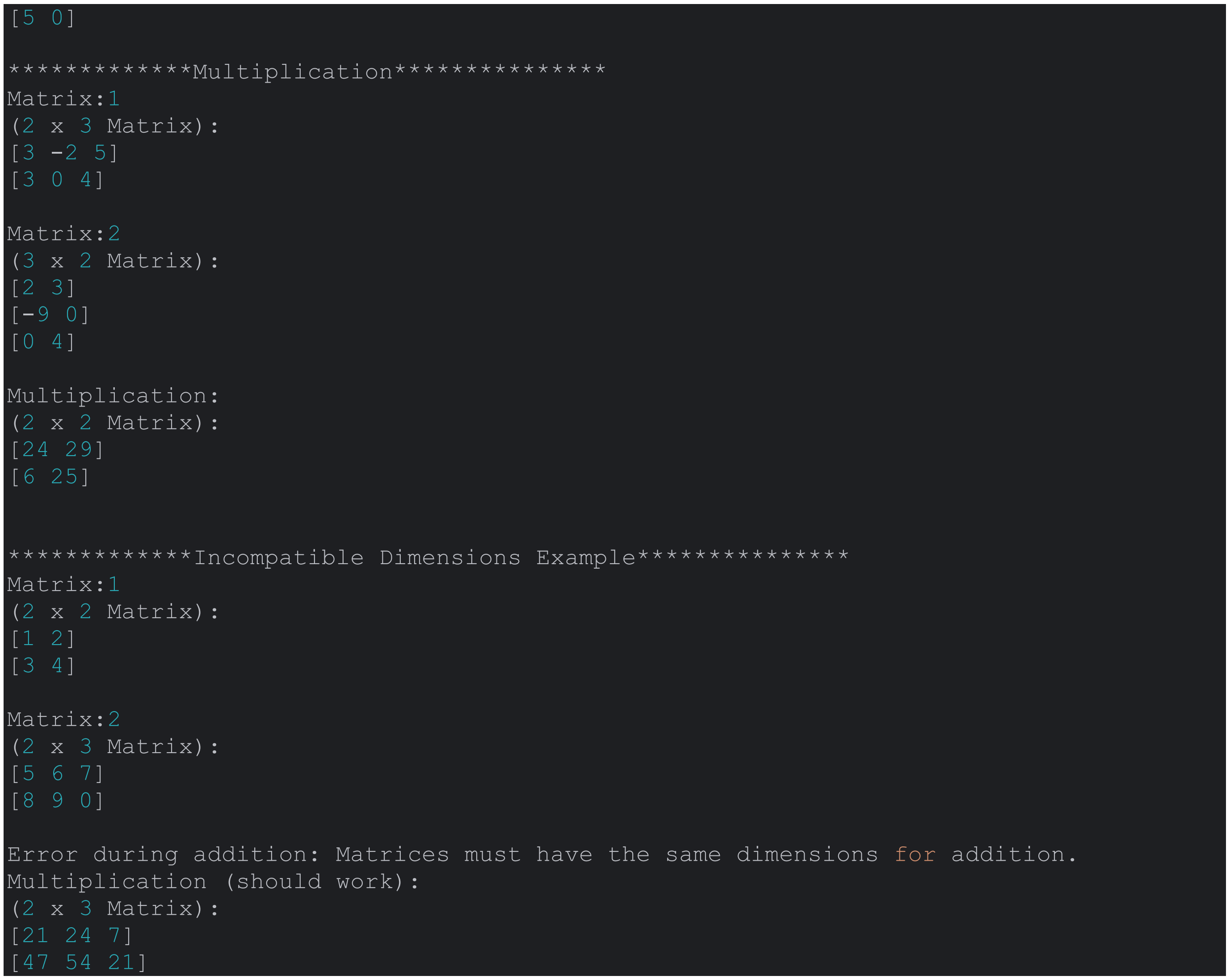
```
class Matrix(val noOfRow: Int, val noOfCol: Int, val data: Array<IntArray>) {

    // Primary constructor for creating a matrix directly with data
    init {
        // Basic validation for matrix dimensions matching provided data
        if (data.size != noOfRow || (noOfRow > 0 && data[0].size != noOfCol)) {
            throw IllegalArgumentException("Matrix dimensions do not match provided
data.")
        }
    }

    operator fun plus(other: Matrix): Matrix {
        if (noOfRow != other.noOfRow || noOfCol != other.noOfCol) {
            throw IllegalArgumentException("Matrices must have the same dimensions
for addition.")
        }

        val resultData = Array(noOfRow) { IntArray(noOfCol) }
        for (i in 0 until noOfRow) {
            for (j in 0 until noOfCol) {
                resultData[i][j] = this.data[i][j] + other.data[i][j]
            }
        }
        return Matrix(noOfRow, noOfCol, resultData)
    }


    operator fun minus(other: Matrix): Matrix {
        if (noOfRow != other.noOfRow || noOfCol != other.noOfCol) {
            throw IllegalArgumentException("Matrices must have the same dimensions
for subtraction.")
        }

        val resultData = Array(noOfRow) { IntArray(noOfCol) }
        for (i in 0 until noOfRow) {
            for (j in 0 until noOfCol) {
                resultData[i][j] = this.data[i][j] - other.data[i][j]
            }
        }
        return Matrix(noOfRow, noOfCol, resultData)
    }

    operator fun times(other: Matrix): Matrix {
        // For multiplication, the number of columns in the first matrix must equal
        // the number of rows in the second matrix.
        if (this.noOfCol != other.noOfRow) {
            throw IllegalArgumentException(
                "Incompatible dimensions for multiplication: " +
                        "Matrix1 columns (${this.noOfCol}) must equal Matrix2 rows
(${other.noOfRow})."
```

```
            )
        }

        val resultData = Array(this.noOfRow) { IntArray(other.noOfCol) }
        for (i in 0 until this.noOfRow) {
            for (j in 0 until other.noOfCol) {
                var sum = 0
                for (k in 0 until this.noOfCol) {
                    sum += this.data[i][k] * other.data[k][j]
                }
                resultData[i][j] = sum
            }
        }
        return Matrix(this.noOfRow, other.noOfCol, resultData)
    }


    override fun toString(): String {
        val sb = StringBuilder()
        sb.append("(${noOfRow} x ${noOfCol} Matrix):\n")
        for (i in 0 until noOfRow) {
            sb.append("[")
            for (j in 0 until noOfCol) {
                sb.append(data[i][j])
                if (j < noOfCol - 1) {
                    sb.append(" ")
                }
            }
            sb.append("]\n")
        }
        return sb.toString()
    }
}

fun main() {
    // Example matrices for addition and subtraction
    val secondMatrix1 = Matrix(
        noOfRow = 3, noOfCol = 2,
        data = arrayOf(
            intArrayOf(6, 3),
            intArrayOf(9, 0),
            intArrayOf(5, 4)
        )
    )

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

    // Example matrices for multiplication
    val firstMatrix = Matrix(
        noOfRow = 2, noOfCol = 3,
        data = arrayOf(
            intArrayOf(3, -2, 5),
            intArrayOf(3, 0, 4)
        )
    )

    val thirdMatrixMultiplication = Matrix( // Naming to avoid conflict with
`thirdMatrix` for addition
        noOfRow = 3, noOfCol = 2,
        data = arrayOf(
            intArrayOf(2, 3),
            intArrayOf(-9, 0),
            intArrayOf(0, 4)
        )
    )


    // --- Addition ---
    println("*************Addition**************")
    println("Matrix:1 \n$secondMatrix1")
    println("Matrix:2 \n$secondMatrix")
    try {
        val additionResult = secondMatrix1 + secondMatrix // Using overloaded '+'
operator
        println("Addition: \n$additionResult")
    } catch (e: IllegalArgumentException) {
        println("Error during addition: ${e.message}")
    }

    // --- Subtraction ---
    println("*************Subtraction**************")
    println("Matrix:1 \n$secondMatrix1")
    println("Matrix:2 \n$secondMatrix")
    try {
        val subtractResult = secondMatrix1 - secondMatrix // Using overloaded '-'
operator
        println("Subtraction: \n$subtractResult")
    } catch (e: IllegalArgumentException) {
        println("Error during subtraction: ${e.message}")
    }

    // --- Multiplication ---
    println("*************Multiplication**************")
    println("Matrix:1 \n$firstMatrix")
    println("Matrix:2 \n$thirdMatrixMultiplication")
    try {
        val multiplicationResult = firstMatrix * thirdMatrixMultiplication // Using
overloaded '*' operator
        println("Multiplication: \n$multiplicationResult")
    } catch (e: IllegalArgumentException) {
        println("Error during multiplication: ${e.message}")
    }

    // Example of incompatible dimensions
    println("\n*************Incompatible Dimensions Example**************")
    val matrixIncompatible1 = Matrix(2, 2, arrayOf(intArrayOf(1,2),
intArrayOf(3,4)))
    val matrixIncompatible2 = Matrix(2, 3, arrayOf(intArrayOf(5,6,7),
intArrayOf(8,9,0)))
```

```
    println("Matrix:1 \n$matrixIncompatible1")
    println("Matrix:2 \n$matrixIncompatible2")
    try {
        val incompatibleAdd = matrixIncompatible1 + matrixIncompatible2
        println("Addition (should fail): \n$incompatibleAdd")
    } catch (e: IllegalArgumentException) {
        println("Error during addition: ${e.message}")
    }

    try {
        val incompatibleMultiply = matrixIncompatible1 * matrixIncompatible2
        println("Multiplication (should work): \n$incompatibleMultiply")
    } catch (e: IllegalArgumentException) {
        println("Error during multiplication: ${e.message}")
    }
}
```

Output:

```
************Addition**************
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
(3 x 2 Matrix):
[2 3]
[-9 0]
[0 4]

Multiplication:
(2 x 2 Matrix):
[24 29]
[6 25]


************Incompatible Dimensions Example**************
Matrix:1
(2 x 2 Matrix):
[1 2]
[3 4]

Matrix:2
(2 x 3 Matrix):
[5 6 7]
[8 9 0]

Error during addition: Matrices must have the same dimensions for addition.
Multiplication (should work):
(2 x 3 Matrix):
[21 24 7]
[47 54 21]
```