[2CEIT5PE18: MOBILE APPLICATION DEVELOPMENT]

Practical: 1

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

Submitted By: Nishu Patel
Enrollment number: 22012011085

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(){
 val a:Int = 5
println("Integer value = $a")
val b:Float = 30.00F
println("Floating value = $b")
val c:Char = 'v'
println("Character value = $c")
val d:String = "Nishu"
println("String value = $d")
val e:Double = 30.56
println("Double value = $e")
val f:Long = 15000000000L
println("Long value = $f")
val g:Short = 5000
println("Short value = $g")
 val h:Boolean = false
println("Boolean value = $h")
val i:Byte = 127
 println("Byte value = $i")
}
```

```
C:\Users\Nishu\.jdks\openjdk
Integer value = 5
Floating value = 30.0
Character value = v
String value = Nishu
Double value = 30.56
Long value = 15000000000
Short value = 5000
Boolean value = false
Byte value = 127
```

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

```
fun main(){
var a : Int = 3
var b: Double = a.toDouble()
println("Integer =$a")
println("Integer to Double = $b")
var s1 : String = "100"
var c : Int = Integer.valueOf(s1)
println("String = $s1")
println("String to Integer = $c")
var s2 : String = "10.89"
var d :Double =s2.toDouble()
println("String=$s2")
println("String to Double = $d")
}
```

```
C:\Users\Nishu\.jdks\openjdk-22.0.
Integer =3
Integer to Double = 3.0
String = 100
String to Integer = 100
String=10.89
String to Double = 10.89

Process finished with exit code 0
```

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("Enter Student Enrollment no:")
val eno = readLine()
println("Enter Student Name:")
val name = readLine()
println("Enter Branch:")
val branch = readLine()
println("Enter Class:")
val classname = readLine()
println("Enter Batch:")
val batch = readLine()
println("Enter CollegeName:")
 val college = readLine()
println("Enter University:")
val uni = readLine()
println("Enter Age:")
val age = readLine()
    println()
    println("**********************")
println()
    println("Student Information")
println("Student Enrollment no: $eno")
```

```
println("Student Name: $name")
println("Student Branch: $branch")
println("Student Class: $classname")
println("Student Batch: $batch") println("Student
College: $college")
println("Student University: $uni")
println("Student Age: $age")
}
```

```
Enter Student Enrollment no:
22012011085
Enter Student Name:
Nishu
Enter Branch:
CE
Enter Class:
5CE-B
Enter Batch:
B4
Enter CollegeName:
UVPCE
Enter University:
GUNI
Enter Age:
20

**********

Student Information
Student Enrollment no: 22012011085
Student Name: Nishu
Student Branch: CE
Student Class: 5CE-B
Student Batch: B4
```

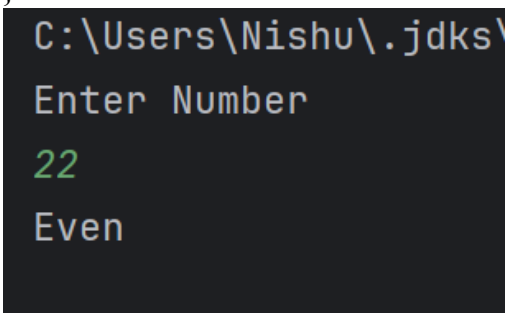
```
**********

Student Information
Student Enrollment no: 22012011085
Student Name: Nishu
Student Branch: CE
Student Class: 5CE-B
Student Batch: B4
Student College: UVPCE
Student University: GUNI
Student Age: 20
```

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

```
fun main(){
   println("Enter Number")
```

```
    val num = readLine()!!.toInt()
    println(
        if (num % 2 == 0)
        {
            "Even"
        }
else
        {
            "Odd"
        }
    )
}
```



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

```
fun main() {    println("Enter the
MonthNumber")
    val MonthNumber = readLine()!!.toInt()

    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 -> "Invalid"
    }
    println(MonthName)
```

}

```
C:\Users\Nishu\.jdks\openjd
Enter the MonthNumber
11
November
```

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

```
fun main() {
println("Enter the first number")
val a = readLine()!!.toInt()
println("Enter the second number")
 val b = readLine()!!.toInt()

   println("Addition of $a and $b = ${add(a,b)}")
println("Subtraction of $a and $b = ${sub(a,b)}")
println("Multiplication of $a and $b = ${mul(a,b)}")
println("Division of $a and $b = ${div(a,b)}")

}

fun add(a:Int,b:Int):Int{
return a+b
}
fun sub(a:Int,b:Int):Int{
   return a-b
}
fun mul(a:Int,b:Int):Int{
   return a*b
}
fun div(a:Int,b:Int):Int{
   return a/b
}
```

```
Enter the first number
6
Enter the second number
3
Addition of 6 and 3 = 9
Subtraction of 6 and 3 = 3
Multiplication of 6 and 3 = 18
Division of 6 and 3 = 2
```

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

```
fun main() {
println("Enter the number")
 val num = readLine()!!.toInt()

   println("factorial of $num = ${fact(num)}")
}
fun fact(num:Int):Int{
   if(num == 0 || num == 1)
   {
      return 1
   }
else
   {
      return num*fact(num-1)
   }

}
```

```
Enter the number
4
factorial of 4 = 24
```

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.

```
import java.util.*

fun main(){
val Array_1 = arrayOf(1,2,3,4,5)
   println(Array_1.joinToString())

   val Array_2 = Array(5){0}
   println(Array_2.joinToString())

   val Array_3 = Array(7) { i -> i}
   println(Array_3.joinToString())

   val Array_4 = IntArray(5){0}
   println(Array_4.joinToString())

   val Array_5 = intArrayOf(5, 2, 8, 3, 9)
for (i in 0..Array_5.lastIndex) {
println(Array_5[i])
   }

   val array2D_1 = arrayOf(
arrayOf(1, 2, 3),
arrayOf(4, 5, 6),
arrayOf(7, 8, 9)
   )
   println(java.util.Arrays.deepToString(array2D_1))

   val array2D_2 = arrayOf(
intArrayOf(1, 2, 3),
```

```
 intArrayOf(4, 5, 6),
 intArrayOf(7, 8, 9)
    )
    println(array2D_2.contentDeepToString())

    val numbers = IntArray(5)
println("please enter the values:")
for (i in 0 until 5) {
print("Enter value ${i + 1}: ")
        numbers[i] = readLine()!!.toInt()
    }
    println("Values using 'until' statement:")
    for (i in 0 until 5) {
        print("${numbers[i]} ")
    }
    println()
println("Values using 'downTo' statement:")
    for (i in 4 downTo 0) {
        print("${numbers[i]} ")
    }


    println("**********Shorting with built in function**********")
    val num = intArrayOf(3, 1, 4, 2, 5)
numbers.sort()
println(numbers.joinToString(", "))

    println("**********Shorting without built in function********")
    val num1 = intArrayOf(3, 1, 4, 2, 5)
for (i in 0 until num1.size - 1) {
for (j in 0 until num1.size - i - 1) {
if (num1[j] > num1[j + 1]) {
// Swap elements
 val temp = num1[j]
num1[j] = num1[j + 1]
num1[j + 1] = temp
            }
        }
    }
```

```
    println(num1.joinToString(", "))


}
```

```
1, 2, 3, 4, 5
0, 0, 0, 0, 0
0, 1, 2, 3, 4, 5, 6
0, 0, 0, 0, 0
5
2
8
3
9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
please enter the values:
Enter value 1: 4
Enter value 2: 3
Enter value 3: 2
Enter value 4: 3
Enter value 5: 2
Values using 'until' statement:
4 3 2 3 2
Values using 'downTo' statement:
2 3 2 3 4 **********Shorting with built in function**********
2, 2, 3, 3, 4
**********Shorting without built in function********
1, 2, 3, 4, 5
1
```

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

```
fun main(){
    val Array_1 = arrayListOf<Int>()
print("enter how many numbers do you want to enter?")
val size = readLine()!!.toInt()
```

```
 if(size>0){
for(i in 1..size){
        print("enter $i number:")
val number = readLine()!!.toInt()
        Array_1.add(number)
      }
      val maxnumber = Array_1.max()
      println("maxnumber = $maxnumber")
   }

}
```

```
enter how many numbers do you want to enter?4
enter 1 number:3
enter 2 number:1
enter 3 number:6
enter 4 number:4
maxnumber = 6
```

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(type: String, model:String, price:Double, currentPrice:Double, owner:
String,milesDriven:Double){
   val type:String
val model:String
val price:Double
 val currentPrice:Double
val owner:String
   val milesDriven:Double
    init {
 this.type = type
this.model = model
```

```
 this.price = price
this.currentPrice= currentPrice
this.owner = owner
 this.milesDriven = milesDriven

    }
    fun showInfo(){
println("Car Information: $type,$model")
println("Car Owner: $owner")
println("Miles Driven: $milesDriven")
println("Original Car Price: $price")
        println("Current Car Price: $currentPrice")
    }
}

fun main(){

    val car1 = Car("BMW","2018",400000.0,399800.0,"Nishu",20.0)
val car2 = Car("Toyota","2019",1080000.0,1079000.0,"Harsh",100.0)

    car1.showInfo()
car2.showInfo()

}
```

```
Car Information: BMW,2018
Car Owner: Nishu
Miles Driven: 20.0
Original Car Price: 400000.0
Current Car Price: 399800.0
Car Information: Toyota,2019
Car Owner: Harsh
Miles Driven: 100.0
Original Car Price: 1080000.0
Current Car Price: 1079000.0
```

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 data:Array<IntArray>,val rows:Int,val cols:Int){

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

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

    // Operator overloading for Matrix subtraction
operator fun minus(other: Matrix): Matrix {
 if (this.rows != other.rows || this.cols != other.cols) {
  throw IllegalArgumentException("Matrices must have the same dimensions for subtraction")
```

```
        }

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

    // Operator overloading for Matrix multiplication
operator fun times(other: Matrix): Matrix {
if (this.cols != other.rows) {
  throw IllegalArgumentException("Number of columns in the first matrix must equal the number of
rows in the second matrix for multiplication")
        }

        val result = Array(this.rows) {
IntArray(other.cols) }
for (i in 0 until this.rows) {
for (j in 0 until other.cols) {
for (k in 0 until this.cols) {
result[i][j] += this.data[i][k] * other.data[k][j]
                }
            }
        }
        return Matrix(result, this.rows, other.cols)
    }

    override fun toString(): String {        val matrixStr =
data.joinToString("\n") { row -> row.joinToString(" ") }        return "($rows
x $cols Matrix):\n$matrixStr"
    }
}


fun main(){

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

```
val secondMatrix = Matrix(arrayOf(intArrayOf(2,3), intArrayOf(-9,0), intArrayOf(0,4)),3,2)
val secondMatrix1 = Matrix(arrayOf(intArrayOf(6,3), intArrayOf(9,0), intArrayOf(5,4)),3,2)
println("**********Addition**********")
   println("Matrix1:")
println("$secondMatrix1")     println("Matrix2:")
println("$secondMatrix")     val AddMatrix =
secondMatrix1 + secondMatrix
println("Addition = $AddMatrix")

   println("**********Subtraction**********")
   println("Matrix1:")     println("$secondMatrix1")
println("Matrix2:")     println("$secondMatrix")     val
SubtractMatrix = secondMatrix1 - secondMatrix
   println("Subtraction = $SubtractMatrix")

   println("**********Multiplication**********")
println("Matrix1:")     println("$firstMatrix")
println("Matrix2:")     println("$secondMatrix")     val MulMatrix
= firstMatrix * secondMatrix
   println("Multiplication = $MulMatrix")

}
```

```
**************Addition*****************
Matrix1:
(3 x 2 Matrix):
6 3
9 0
5 4
Matrix2:
(3 x 2 Matrix):
2 3
-9 0
0 4
Addition = (3 x 2 Matrix):
8 6
0 0
5 8
**************Subtraction*****************
Matrix1:
(3 x 2 Matrix):
6 3
9 0
5 4
Matrix2:
(3 x 2 Matrix):
2 3
-9 0
```

```
Matrix2:
(3 x 2 Matrix):
2 3
-9 0
0 4
Subtraction = (3 x 2 Matrix):
4 0
18 0
5 0
**************Multiplication*****************
Matrix1:
(2 x 3 Matrix):
3 2 -5
3 0 4
Matrix2:
(3 x 2 Matrix):
2 3
-9 0
0 4
Multiplication = (2 x 2 Matrix):
-12 -11
6 25

Process finished with exit code 0
```