[2CEIT5PE18: MOBILE APPLICATION DEVELOPMENT]

Practical: 1

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

Submitted By: Makavana Raj Mahendrabhai Enrollment number: 22012011068

Departmentof 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(){

```
val a : Int = 22 val b : Double
= 96.36 val c : Float = 1.5f
val d : Long = 785214698
val e : Short = 45 val f : Byte
= 13 val g : Char = 'K' val h :
Boolean = true val i : String
= "Raj"
println ("Integer Value : $a")
println ("Double Value : $b")
println ("Float Value : $c")
println ("Long Value : $d")
println ("Short Value : $e")
println ("Byte Value : $f")
println ("Char Value : $g")
println ("Boolean Value : $h")
println ("String Value : $i")
}
```

Output:

```
Integer Value : 22
Double Value : 96.36
Float Value : 1.5
Long Value : 785214698
Short Value : 45
Byte Value : 13
Char Value : K
Boolean Value : true
String Value : Raj
```

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

Answer: fun

```
main(){
    val a = 10
    println("Integer value : $a")
    val b = a.toDouble()
    println("Double value (From Int) : $b")
    val c = "10"
    println("String value : $c")
    val d = c.toInt()
    println("Integer value (From String) : $d")
     val e = "10.2"
    val f = e.toDouble()
```

```
    println("Double value (From String) : $f")
}
```

Output:

```
Integer value : 10
Double value (From Int) : 10.0
String value : 10
Integer value (From String) : 10
Double value (From String) : 10.2
```

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(){
    print("Student E.N.no : ")
    val enno = readln()
    print("Student Name : ")
    val name = readln()
    print("Student Branch : ")
    val branch = readln()
    print("Student Class : ")
    val clas = readln()
    print("Student Batch : ")
    val batch = readln()
    ("Student Age : ")
    val age = readln()
    print()
    print("**************************")
    print()
    print("Student's Data")
    print("E.N.no : $enno")
    print("Name : $name")
    print("Age : $age")
    print("Branch : $branch")
    print("Class : $clas")
    print("Batch : $batch")
}
```

Output:

```
Student E.N.no : 22012011068
Student Name : Raj
Student Branch : CE
Student Class : B
Student Batch : B4
Student Age : 19

**************************

Student's Data
E.N.no : 22012011068
Name : Raj
Age : 19
Branch : CE
Class : B
Batch : B4
```

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

Answer: fun
main(){
print("Enter a number : ") val
num = readln()toInt()
println(
if (num % 2 == 0){
"Number $num is even number"
}
else{
"Number $num is odd number"
}
)
}

Output:

```
Enter a number : 5
Number 5 is odd number
```

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

Answer:

```
fun main(){
    print("Enter Month Number : ")
    val month = readln().toInt()
    when(month){
        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 proper Number")

}
```

Output:

```
Enter Month Number : 5
May
```

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

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

```
}
fun div(a : Int,b : Int) : Int{ return
    a/b
}
fun main(){
    val  a  =  111
    val  b  =  2222
    val c = -222
    println ("Addition of $a, $b, $c is ${ add(a,b,c)}")
    println ("Subtraction of $a, $b, $c is ${ sub(a,b,c)}")
    println ("Multiplication of $a, $b, $c is "+ mul(a,b,c))
    println ("Division of $b, $a is "+div(b,a))
}
```

Output:

```
Addition of 111, 2222, -222 is 2111
Subtraction of 111, 2222, -222 is -1889
Multiplication of 111, 2222, -222 is -54754524
Division of 2222, 111 is 20
```

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

```
fun fact(num:Int):Int {
    return when (num){
        0 -> 1
        1 -> 1
        else -> num*fact(num-1)
    }
}
fun main(){
    print("Enter Number : ")
    val num = readln().toInt()
    print("Factorial of $num is : "+fact(num))
}
```

Output:

```
Enter Number : 5
Factorial of 5 is : 120
```

1.8 Working with Arrays: Explore array operationssuch 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:

```
fun main(){
    val array1 = arrayOf(1,2,3)
    println("Using arrayOf() method : ${array1.contentToString()}") val
    array2 = arrayOf<Int>(3,4,5)
    println("Using arrayOf<>() method : ${array2.contentToString()}")
    val array3 = Array<Int>(7) { i -> i * 1 }
    println("Using Array<>(){} method : ") for
    (i in 0..array3.size-1)
    {
        print("${array3[i]} ")
    }
    println ()
    println ("Using IntArray(){} method : ") val
    array4 = IntArray(3) { i -> i * 2 } for (i in
    0..array4.size-1)
    {
        print("${array4[i]} ")
    }
    println()
    val array5 = intArrayOf(1,2,3)
    println("Using IntArray() method : ${array5.joiToString()}")
    val  array6 = arrayOf(
        arrayOf  (1,2,3),
        arrayOf  (3,4,5),
        arrayOf  (5,6,7)
    )
    println("2D Array : ${array6. contentDeepToString(()}")
    val array7 = IntArray(5) for
    (i in array7.indices){
        print("Enter value of array7[$i] : ")
        array7[i] = readln().toInt()
    }
    println("Your array : ${array7.joinToString()}") array7.
      ()      sort
    println("Using builtIn fun. sort : ${array7.joinToString()}") for
    (i in array7.
```

```
    for (j in array7. indices){    ){
        if                  indices
            (array7[i]<array7[j]){
            val temp = array7[i]
            array7[i] = array7[j]
            array7[j] = temp
        }
    }
    println("Without Using builtIn fun. sort : ${array7.joinToString()}")
}
```

Output:

```
Using arrayOf() method : [1, 2, 3]
Using arrayOf<>() method : [3, 4, 5]
Using Array<>(){} method :
0 1 2 3 4 5 6
Using IntArray(){} method :
0 2 4
Using IntArray() method : 1, 2, 3
2D Array : [[1, 2, 3], [3, 4, 5], [5, 6, 7]]
Enter value of array7[0] : 4
Enter value of array7[1] : 2
Enter value of array7[2] : 1
Enter value of array7[3] : 5
Enter value of array7[4] : 3
Your array : 4, 2, 1, 5, 3
Using builtIn fun. sort : 1, 2, 3, 4, 5
Without Using builtIn fun. sort : 1, 2, 3, 4, 5
```

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

Answer:

```
fun main() {
val a = IntArray(5) for
(i in 0..4) {
        print("Enter value of a[$i] : ")
        a[i] = readln().toInt()
    }
    val max = a.max()
    println("Max value is $max")
}
```

Output:

```
Enter value of a[0] : 4
Enter value of a[1] : 2
Enter value of a[2] : 3
Enter value of a[3] : 5
Enter value of a[4] : 1
Max value is 5
```

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:

```
class Car (val info : String, val owner : String, val miles : Int, val oPrice : Int, val cPrice : Int) { fun
    information() {
        println ("**********************************")
        println ("Car information : $info")
        println ("Car owner : $owner")
        println ("Miles Drive : $miles")
        println ("Original Car Price : $oPrice")
        println ("Current Car Price : $cPrice")
        println
        ("**********************************")
        ()
    }
}
fun main(){
    val c1 = Car("BMW, 2015", "Kirtan", 105, 100000, 98950)
    val c2 = Car("BMW, 2019","Keyur",20,400000,399800) val
    c3 = Car("Toyota","Kartik",100,10880000,1079000) val c4 =
    Car("Maruti","Karan",200,400000,399800) c1.information()
    c2.information() c3.information() c4.information()}
```

Output

```
************************************
Car information : BMW, 2015
Car owner : Raj
Miles Drive : 105
Original Car Price : 100000
Current Car Price : 98950
************************************
************************************
Car information : BMW, 2019
Car owner : Aman
Miles Drive : 20
Original Car Price : 400000
Current Car Price : 399800
************************************
************************************
Car information : Toyota
Car owner : Vasant
Miles Drive : 100
Original Car Price : 10880000
Current Car Price : 1079000
************************************
************************************
Car information : Maruti
Car owner : rahul
Miles Drive : 200
Original Car Price : 400000
Current Car Price : 399800
************************************
```

:

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. Answer:

```
class Matrix(private val data: Array<IntArray>, val rows: Int, val cols: Int) {
    override fun toString(): String { val result = StringBuilder() for (row in data) {
            result.append(row.joinToString(" ", "[", "]")).append("\n")
    }
    return result.toString()
}
operator fun plus(other: Matrix): Matrix { if (this.rows != other.rows || this.cols != other.cols)
    { throw IllegalArgumentException("Matrices dimensions do not match for addition")


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

```
    }
    return Matrix(result, rows, cols)
}
operator fun minus(other: Matrix): Matrix {
    if (this.rows != other.rows || this.cols != other.cols) {
        throw IllegalArgumentException("Matrices dimensions do not match 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 fun times(other: Matrix): Matrix {
    if (this.cols != other.rows) {
        throw IllegalArgumentException("Matrices dimensions do not match for multiplication")
    }
    val result = Array(this.rows) { IntArray(other.cols) } for (i
    in 0    this until .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)
    }
}



fun main() {
    val firstMatrix = Matrix(arrayOf(intArrayOf(5, -2, 5), intArrayOf(5, 0, 4)), 2, 3)
    val secondMatrix = Matrix(arrayOf(intArrayOf(2, 3), intArrayOf(-9, 6), intArrayOf(0, 4)), 3, 2)
    val secondMatrix1 = Matrix(arrayOf(intArrayOf(6, 3), intArrayOf(9, 6), intArrayOf(5, 4)), 3, 2)
```

```
println ("**************** Addition ****************")
println ("Matrix: 1 ")
println (secondMatrix1)
println ("Matrix: 2 ")
println (secondMatrix) val thirdMatrix =
secondMatrix1 + secondMatrix
println ("Addition: $thirdMatrix")
println ("**************** Subtraction ****************")
println ("Matrix: 1 ")
println (secondMatrix1)
println ("Matrix: 2 ")
println (secondMatrix) val subtractMatrix =
secondMatrix1 - secondMatrix
println ("Subtraction: $subtractMatrix")
println ("**************** Multiplication ****************")
println ("Matrix: 1 ")
println (firstMatrix)
println ("Matrix: 2 ")
println (secondMatrix) val multiplication =
firstMatrix * secondMatrix
println("Multiplication: \n$multiplication")
}
```

Output:

```
********** Addition **********
Matrix: 1
[6 3]
[9 6]
[5 4]

Matrix: 2
[2 3]
[-9 6]
[0 4]

Addition: [8 6]
[0 12]
[5 8]

********** Subtraction **********
Subtraction: [4 0]
[18 0]
[5 0]

********** Multiplication **********
Multiplication:
[28 23]
[10 31]
```