

LAB 3

Q1. Scala program to implement the foreach loop on a list of numbers.

Ans: **Code:-**

```
HelloWorld.scala 42
1 object ForEach extends App {
2
3   val numbers = List(1, 2, 3, 4, 5,6,7)
4
5
6   numbers.foreach(num => println(num))
7   //vaswati//
8 }
9
```

OUTPUT:

NEW SCALA RUN

STDIN

Input for the program (Optional)

Output:

1
2
3
4
5
6
7

Q2. Scala program to create a user define function to return largest number among two numbers entered by user.

Ans: **Code:-**

LAB 3

HelloWorld.scala

```
1 object Func{
2   def comp(a:Int, b:Int): Int={
3     if(a>b){
4       return a;
5     }
6   } else{
7     return b;
8   }
9   }
10 def main(args: Array[String]):Unit={
11   println("Enter the two nos.");
12   var p=scala.io.StdIn.readLine().toInt
13   var q=scala.io.StdIn.readLine().toInt
14   println("The largest no. is " + comp(p,q));
15 }
16 }
17 //vaswati//
```

OUTPUT:-

NEW SCALA RUN

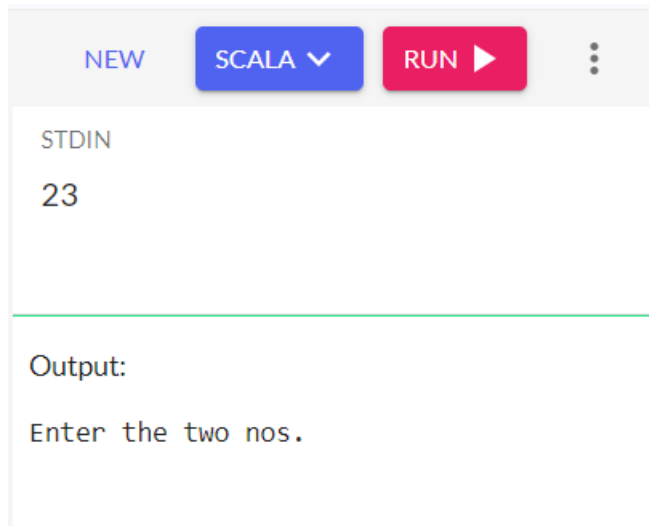
STDIN

10
23

Output:

Enter the two nos.

LAB 3



Q3. Scala code to create a function with default arguments. Wish good morning to the person.

Ans: **Code:-**

```
HelloWorld.scala 423wat
1 ▾ object Night {
2 ▾   def main(args: Array[String]): Unit = {
3     greet()
4     println("Enter name:")
5     val name = scala.io.StdIn.readLine().toString()
6     greet(name)
7   }
8
9 ▾   def greet(name: String="po"): Unit = {
10    println("Hi, " + name + "!")
11  }
12 }
13 //vaswati//
14
```

OUTPUT:-

The screenshot shows a Scala REPL interface. At the top, there are buttons for 'NEW', 'SCALA' (with a dropdown arrow), and 'RUN' (with a play icon). Below these buttons, the 'STDIN' section shows the input 'vaswati' with a red squiggly underline. The 'Output' section shows the following text: 'Hi, po!', 'Enter name:', and 'Hi, vaswati!'.

Q4. Scala code to create anonymous functions for add, sub, and mul with => operator.

Ans: **Code:-**

The screenshot shows a Scala code editor with a file named 'HelloWorld.scala' and a line number '423wat7'. The code defines an object 'AnonymousFunctions' with a 'main' method. Inside the 'main' method, three anonymous functions are defined using the '=>' operator: 'add', 'sub', and 'mul'. These functions are then used to calculate 'resultAdd', 'resultSub', and 'resultMul' for the values 10 and 5. Finally, the results are printed using 'println'.

```

1 object AnonymousFunctions {
2   def main(args: Array[String]): Unit = {
3
4     val add = (x: Int, y: Int) => x + y
5     val sub = (x: Int, y: Int) => x - y
6     val mul = (x: Int, y: Int) => x * y
7
8
9     val resultAdd = add(10, 5)
10    val resultSub = sub(10, 5)
11    val resultMul = mul(10, 5)
12
13
14    println("Addition result: " + resultAdd)
15    println("Subtraction result: " + resultSub)
16    println("Multiplication result: " + resultMul)
17  }
18 }
19 //vaswati//

```

OUTPUT:-

The screenshot shows a Scala IDE interface. At the top, there are buttons for 'NEW', 'SCALA' (with a dropdown arrow), 'RUN' (with a play icon), and a menu icon (three dots). Below these buttons, the text 'STDIN' is visible. A horizontal line separates the header from the output area. In the output area, the text 'Output:' is followed by three lines of results: 'Addition result: 15', 'Subtraction result: 5', and 'Multiplication result: 50'.

Q5. Scala code to create anonymous functions for add, sub, and mul with `_` operator.

Ans: **Code:-**

The screenshot shows a Scala code editor with a file named 'HelloWorld.scala' and a user identifier '423wat7'. The code is as follows:

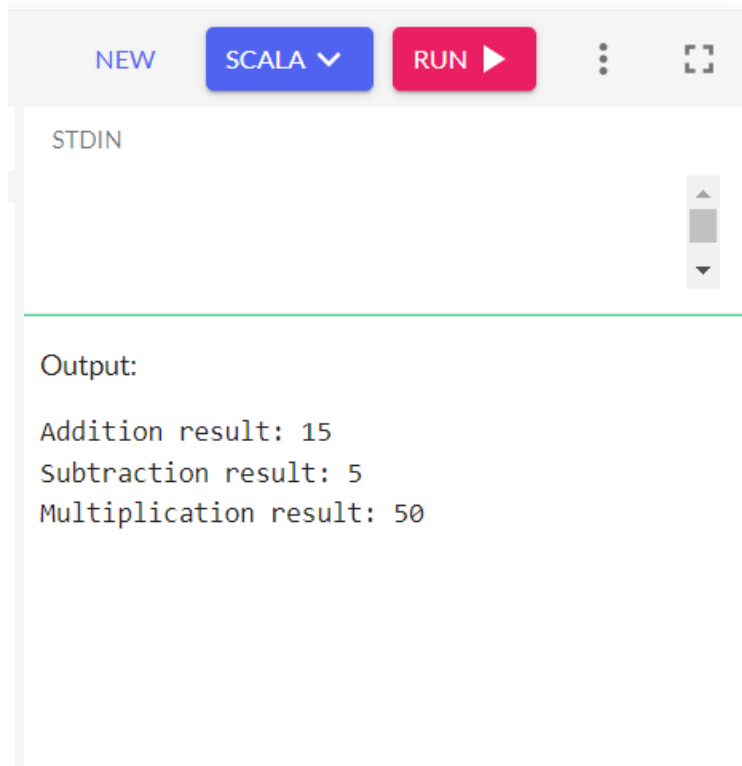
```

1 object AnonymousFunctions {
2   def main(args: Array[String]): Unit = {
3
4     val add = (_: Int) + (_: Int)
5     val sub = (_: Int) - (_: Int)
6     val mul = (_: Int) * (_: Int)
7
8
9     val resultAdd = add(10, 5)
10    val resultSub = sub(10, 5)
11    val resultMul = mul(10, 5)
12
13
14    println("Addition result: " + resultAdd)
15    println("Subtraction result: " + resultSub)
16    println("Multiplication result: " + resultMul)
17  }
18 }
19
20 //vaswati//

```

LAB 3

OUTPUT:-



The screenshot shows a Scala IDE interface. At the top, there is a toolbar with buttons for 'NEW', 'SCALA' (with a dropdown arrow), 'RUN' (with a play icon), and a menu icon (three vertical dots). Below the toolbar is a text area labeled 'STDIN'. A horizontal line separates the 'STDIN' area from the 'Output' area. The 'Output' area displays the following text:

```
Output:  
Addition result: 15  
Subtraction result: 5  
Multiplication result: 50
```