

Lab 1

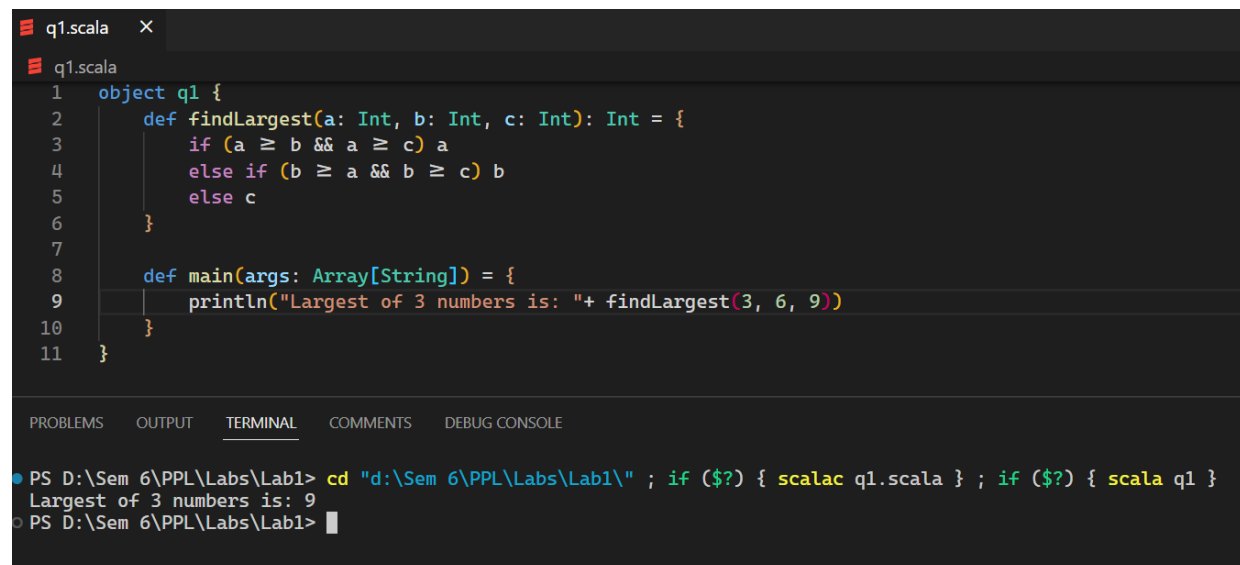
Question 1:

Write a program in Scala to implement a function to find the largest of 3 numbers.

Code:

```
object q1 {  
  def findLargest(a: Int, b: Int, c: Int): Int = {  
    if (a >= b && a >= c) a  
    else if (b >= a && b >= c) b  
    else c  
  }  
  
  def main(args: Array[String]) = {  
    println("Largest of 3 numbers is: " + findLargest(3, 6, 9))  
  }  
}
```

Output:



The screenshot shows an IDE window titled 'q1.scala'. The code is as follows:

```
1 object q1 {  
2   def findLargest(a: Int, b: Int, c: Int): Int = {  
3     if (a >= b && a >= c) a  
4     else if (b >= a && b >= c) b  
5     else c  
6   }  
7  
8   def main(args: Array[String]) = {  
9     println("Largest of 3 numbers is: " + findLargest(3, 6, 9))  
10  }  
11 }
```

Below the code editor, the 'TERMINAL' tab is active, showing the command prompt output:

```
PS D:\Sem 6\PPL\Labs\Lab1> cd "d:\Sem 6\PPL\Labs\Lab1\" ; if ($?) { scalac q1.scala } ; if ($?) { scala q1 }  
Largest of 3 numbers is: 9  
PS D:\Sem 6\PPL\Labs\Lab1>
```

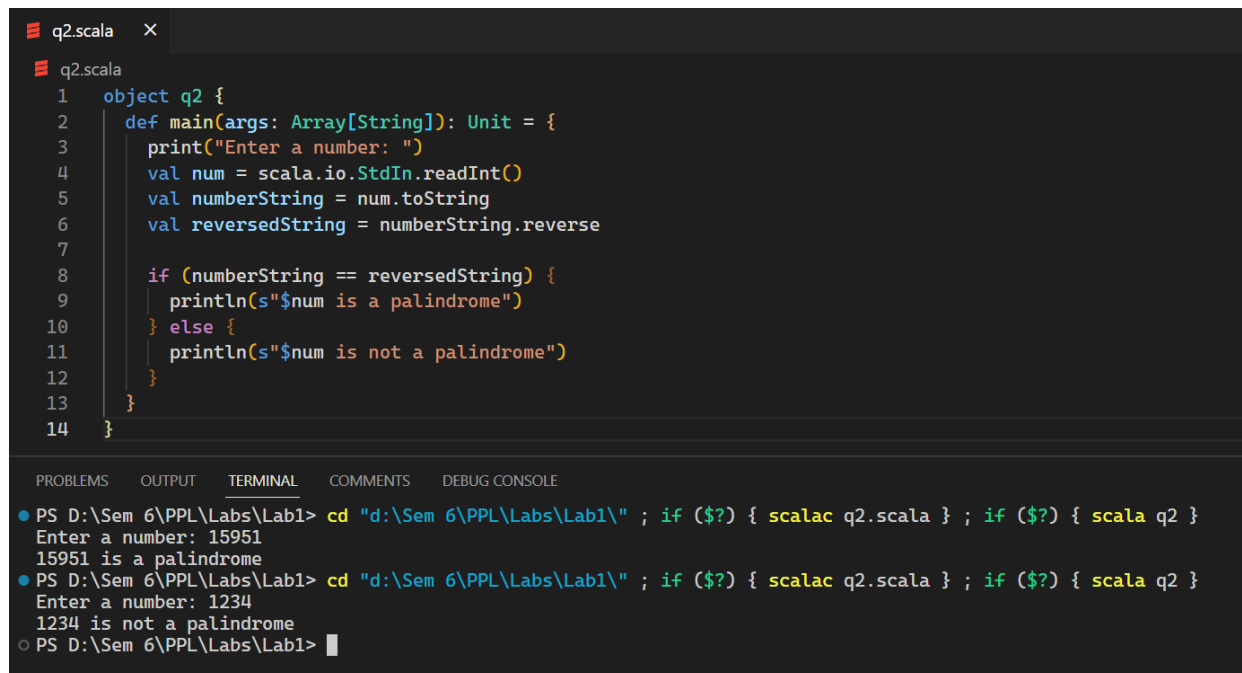
Question 2:

Write a program in Scala to check whether a number entered of user's choice is palindrome or not.

Code:

```
object q2 {  
  def main(args: Array[String]): Unit = {  
    print("Enter a number: ")  
    val num = scala.io.StdIn.readInt()  
    val numberString = num.toString  
    val reversedString = numberString.reverse  
  
    if (numberString == reversedString) {  
      println(s"$num is a palindrome")  
    } else {  
      println(s"$num is not a palindrome")  
    }  
  }  
}
```

Output:



The screenshot shows an IDE with a file named `q2.scala`. The code is identical to the one provided in the previous block. Below the code editor, there is a terminal window with the following output:

```
PS D:\Sem 6\PPL\Labs\Lab1> cd "d:\Sem 6\PPL\Labs\Lab1\" ; if ($?) { scalac q2.scala } ; if ($?) { scala q2 }  
Enter a number: 15951  
15951 is a palindrome  
PS D:\Sem 6\PPL\Labs\Lab1> cd "d:\Sem 6\PPL\Labs\Lab1\" ; if ($?) { scalac q2.scala } ; if ($?) { scala q2 }  
Enter a number: 1234  
1234 is not a palindrome  
PS D:\Sem 6\PPL\Labs\Lab1>
```

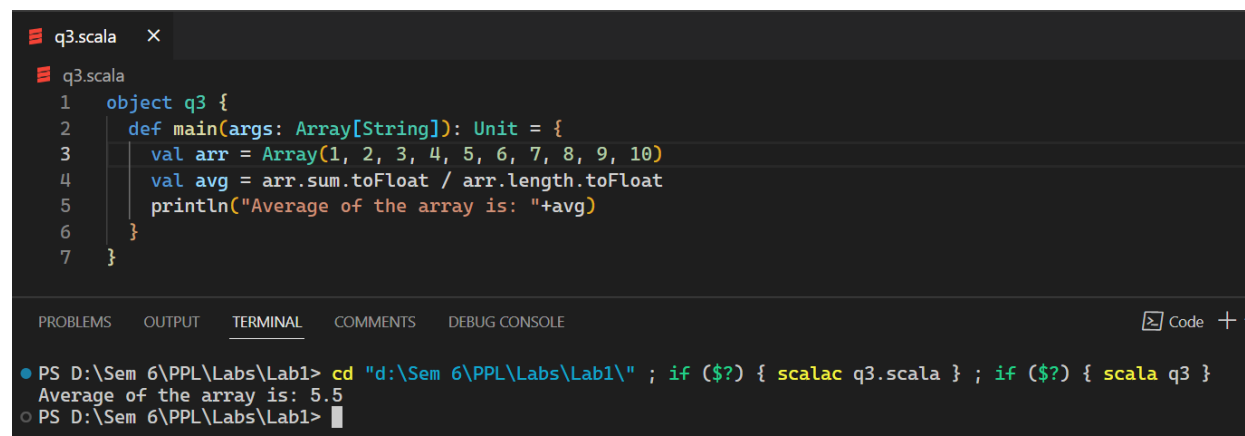
Question 3:

Write a program in Scala to find the average of the array.

Code:

```
object q3 {  
  def main(args: Array[String]): Unit = {  
    val arr = Array(1, 2, 3, 4, 5, 6, 7, 8, 9)  
    val avg = arr.sum.toFloat / arr.length.toFloat  
    println("Average of the array is: "+avg)  
  }  
}
```

Output:



The screenshot shows an IDE with a file named `q3.scala`. The code is as follows:

```
1 object q3 {  
2   def main(args: Array[String]): Unit = {  
3     val arr = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
4     val avg = arr.sum.toFloat / arr.length.toFloat  
5     println("Average of the array is: "+avg)  
6   }  
7 }
```

The terminal output shows the command to compile and run the program, and the resulting output:

```
PS D:\Sem 6\PPL\Labs\Lab1> cd "d:\Sem 6\PPL\Labs\Lab1\" ; if ($?) { scalac q3.scala } ; if ($?) { scala q3 }  
Average of the array is: 5.5  
PS D:\Sem 6\PPL\Labs\Lab1>
```

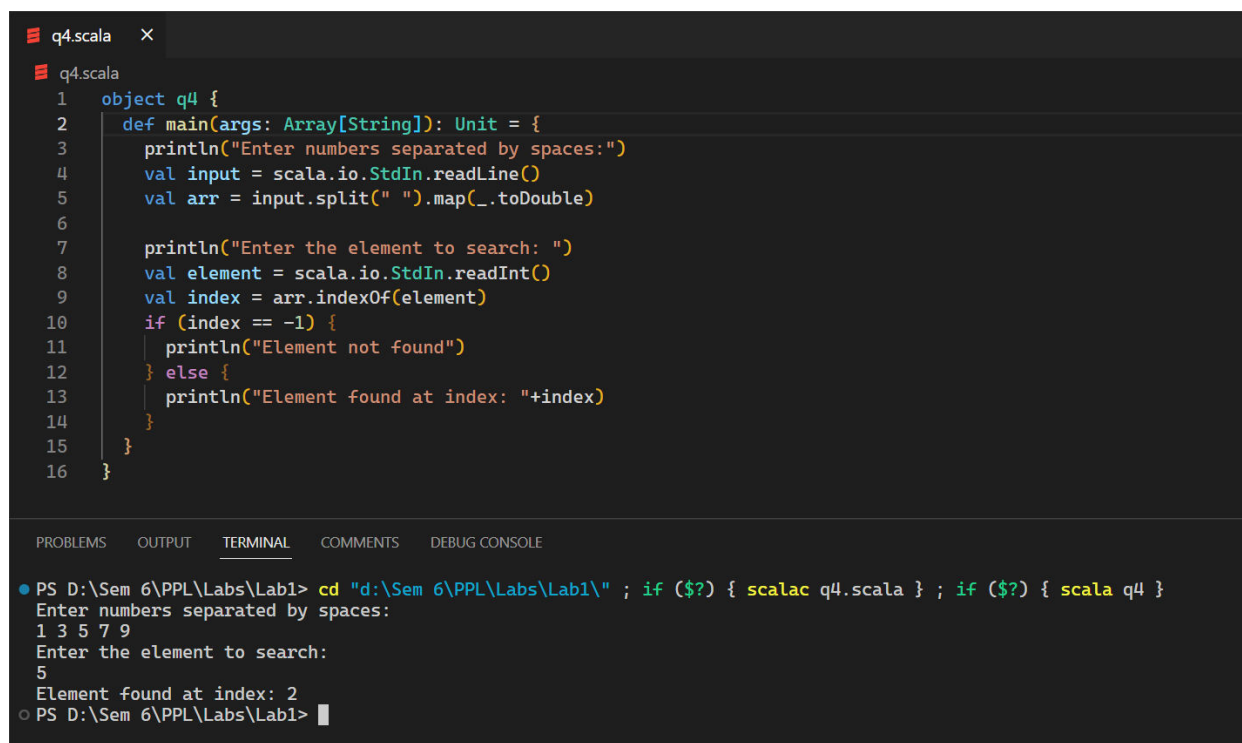
Question 4:

Write a program in Scala to read an array and show the usage of indexOf() function

Code:

```
object q4 {  
  def main(args: Array[String]): Unit = {  
    println("Enter numbers separated by spaces:")  
    val input = scala.io.StdIn.readLine()  
    val arr = input.split(" ").map(_.toDouble)  
  
    println("Enter the element to search: ")  
    val element = scala.io.StdIn.readInt()  
    val index = arr.indexOf(element)  
    if (index == -1) {  
      println("Element not found")  
    } else {  
      println("Element found at index: "+index)  
    }  
  }  
}
```

Output:



```
q4.scala X  
q4.scala  
1  object q4 {  
2    def main(args: Array[String]): Unit = {  
3      println("Enter numbers separated by spaces:")  
4      val input = scala.io.StdIn.readLine()  
5      val arr = input.split(" ").map(_.toDouble)  
6  
7      println("Enter the element to search: ")  
8      val element = scala.io.StdIn.readInt()  
9      val index = arr.indexOf(element)  
10     if (index == -1) {  
11       println("Element not found")  
12     } else {  
13       println("Element found at index: "+index)  
14     }  
15   }  
16 }  
  
PROBLEMS  OUTPUT  TERMINAL  COMMENTS  DEBUG CONSOLE  
● PS D:\Sem 6\PPL\Labs\Lab1> cd "d:\Sem 6\PPL\Labs\Lab1\" ; if ($?) { scalac q4.scala } ; if ($?) { scala q4 }  
Enter numbers separated by spaces:  
1 3 5 7 9  
Enter the element to search:  
5  
Element found at index: 2  
○ PS D:\Sem 6\PPL\Labs\Lab1> █
```

Question 5:

Write a program in Scala to create an array with different colors of rainbow.
Remove specific colors and display the resultant array.

Code:

```
object q5 {  
  def main(args: Array[String]): Unit = {  
    val rainbow = Array("Red", "Orange", "Yellow", "Green", "Blue",  
"Indigo", "Violet")  
    println("Rainbow colors: " + rainbow.mkString(", "))  
    val rainbow1 = rainbow.diff(Array("Red", "Blue", "Green"))  
    println("Rainbow colors after removing Red, Blue and Green: " +  
rainbow1.mkString(", "))  
  }  
}
```

Output:



The screenshot shows an IDE window titled 'q5.scala'. The code is as follows:

```
1 object q5 {  
2   def main(args: Array[String]): Unit = {  
3     val rainbow = Array("Red", "Orange", "Yellow", "Green", "Blue", "Indigo", "Violet")  
4     println("Rainbow colors: " + rainbow.mkString(", "))  
5     val rainbow1 = rainbow.diff(Array("Red", "Blue", "Green"))  
6     println("Rainbow colors after removing Red, Blue and Green: " + rainbow1.mkString(", "))  
7   }  
8 }
```

The terminal output at the bottom shows the execution results:

```
PS D:\Sem 6\PPL\Labs\Lab1> cd "d:\Sem 6\PPL\Labs\Lab1\" ; if ($?) { scalac q5.scala } ; if ($?) { scala q5 }  
Rainbow colors: Red, Orange, Yellow, Green, Blue, Indigo, Violet  
Rainbow colors after removing Red, Blue and Green: Orange, Yellow, Indigo, Violet  
PS D:\Sem 6\PPL\Labs\Lab1>
```

Lab 2

Question 1:

Is List Mutable or not? Prove

Answer:

In scala, lists are immutable. It means that we cannot change the values or size of the lists once they have been created. In the code snippet written below, we can see that the code shows error when we try to modify the list element but it executes successfully when we comment out the line trying to modify the list element.

Code:

```
object q1 {  
  def main(args: Array[String]): Unit = {  
    var alphabets = List('a', 'b', 'c', 'd');  
    println(alphabets);  
    lang(2) = 'e';  
  }  
}
```

Outputs:



The screenshot shows an IDE window titled 'q1.scala'. The code in the editor is:

```
1 object q1 {  
2   def main(args: Array[String]): Unit = {  
3     var alphabets = List('a', 'b', 'c', 'd');  
4     println(alphabets);  
5     // lang(2) = 'e';  
6   }  
7 }
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
PS D:\Sem 6\PPL\Labs\Lab3> cd "d:\Sem 6\PPL\Labs\Lab3\" ; if ($?) { scalac -explain q1.scala } ; if ($?) { scala q1 }  
List(a, b, c, d)  
PS D:\Sem 6\PPL\Labs\Lab3>
```

Name - Shubham Garg
Reg No – BL.EN.U4CSE20158

```
q1.scala x
q1.scala
1 object q1 {
2   def main(args: Array[String]): Unit = {
3     var alphabets = List('a', 'b', 'c', 'd');
4     println(alphabets);
5     lang(2) = 'e';
6   }
7 }

PROBLEMS OUTPUT TERMINAL COMMENTS DEBUG CONSOLE
PS D:\Sem 6\PPL\Labs\Lab3> cd "d:\Sem 6\PPL\Labs\Lab3\" ; if ($?) { scalac -explain q1.scala } ; if ($?) { scala q1 }
-- [E006] Not Found Error: q1.scala:5:4 -----
5 |     lang(2) = 'e';
  |     ^^^^
  |     Not found: lang
  |
  | Explanation (enabled by '-explain')
  | -----
  | The identifier for `lang` is not bound, that is,
  | no declaration for this identifier can be found.
  | That can happen, for example, if `lang` or its declaration has either been
  | misspelt or if an import is missing.
  | -----
1 error found
PS D:\Sem 6\PPL\Labs\Lab3>
```

Question 2:

Is Tuple Mutable or not? Prove

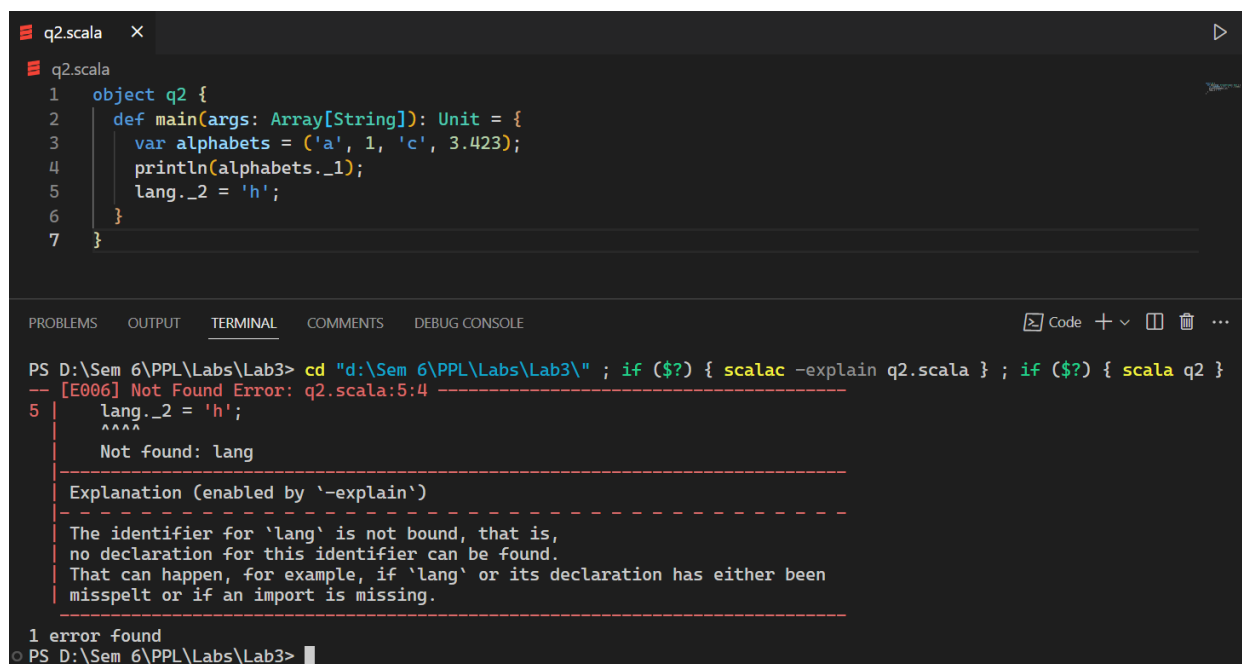
Answer:

In scala, tuples are immutable. It means that we cannot change the values or size of the tuples once they have been created. In the code snippet written below, we can see that the code shows error when we try to modify the tuple element but it executes successfully when we comment out the line trying to modify the tuple element. However, tuples are different from lists as all the elements of the lists have to be of the same data type but in tuple, we can have elements of different data types.

Code:

```
object q2 {  
  def main(args: Array[String]): Unit = {  
    var alphabets = ('a', 1, 'c', 3.423);  
    println(alphabets._1);  
    lang._2 = 'h';  
  }  
}
```

Outputs:



```
q2.scala x  
q2.scala  
1 object q2 {  
2   def main(args: Array[String]): Unit = {  
3     var alphabets = ('a', 1, 'c', 3.423);  
4     println(alphabets._1);  
5     lang._2 = 'h';  
6   }  
7 }
```

PROBLEMS OUTPUT TERMINAL COMMENTS DEBUG CONSOLE


PS D:\Sem 6\PPL\Labs\Lab3> cd "d:\Sem 6\PPL\Labs\Lab3\" ; if (\$?) { scalac -explain q2.scala } ; if (\$?) { scala q2 }

-- [E006] Not Found Error: q2.scala:5:4 -----
5 | lang._2 = 'h';
 | ^^^^^
Not found: lang
Explanation (enabled by '-explain')

The identifier for 'lang' is not bound, that is,
no declaration for this identifier can be found.
That can happen, for example, if 'lang' or its declaration has either been
misspelt or if an import is missing.

1 error found
PS D:\Sem 6\PPL\Labs\Lab3>

Name - Shubham Garg
Reg No – BL.EN.U4CSE20158



The image shows a screenshot of an IDE with a Scala file named `q2.scala` open. The code defines an object `q2` with a `main` function that takes an array of strings and prints the first element. The terminal shows the execution of `scalac` and `scala` commands, resulting in the output `a`.

```
q2.scala
1 object q2 {
2   def main(args: Array[String]): Unit = {
3     var alphabets = ('a', 1, 'c', 3.423);
4     println(alphabets._1);
5     // lang._2 = 'h';
6   }
7 }
```

PROBLEMS OUTPUT TERMINAL COMMENTS DEBUG CONSOLE

PS D:\Sem 6\PPL\Labs\Lab3> cd "d:\Sem 6\PPL\Labs\Lab3\" ; if (\$?) { scalac -explain q2.scala } ; if (\$?) { scala q2 }
a
PS D:\Sem 6\PPL\Labs\Lab3>

Question 3:

Write a Scala program to remove a specific element from an given array.

Code:

```
object q3 {  
  def main(args: Array[String]): Unit = {  
    var arr = Array(1, 2, 3, 4, 5)  
    var toRemove = 3  
    println("Original array: " + arr.mkString(", "))  
    println("Element to remove: " + toRemove)  
    var newArr = arr.filter(_ != toRemove)  
    println("New array: " + newArr.mkString(", "))  
  }  
}
```

Output:



The screenshot shows an IDE window titled 'q3.scala'. The code is as follows:

```
1 object q3 {  
2   def main(args: Array[String]): Unit = {  
3     var arr = Array(1, 2, 3, 4, 5)  
4     var toRemove = 3  
5     println("Original array: " + arr.mkString(", "))  
6     println("Element to remove: " + toRemove)  
7     var newArr = arr.filter(_ != toRemove)  
8     println("New array: " + newArr.mkString(", "))  
9   }  
10 }
```

The terminal output at the bottom shows the command execution and the resulting output:

```
PS D:\Sem 6\PPL\Labs\Lab3> cd "d:\Sem 6\PPL\Labs\Lab3\" ; if ($?) { scalac q3.scala } ; if ($?) { scala q3 }  
Original array: 1, 2, 3, 4, 5  
Element to remove: 3  
New array: 1, 2, 4, 5  
PS D:\Sem 6\PPL\Labs\Lab3>
```

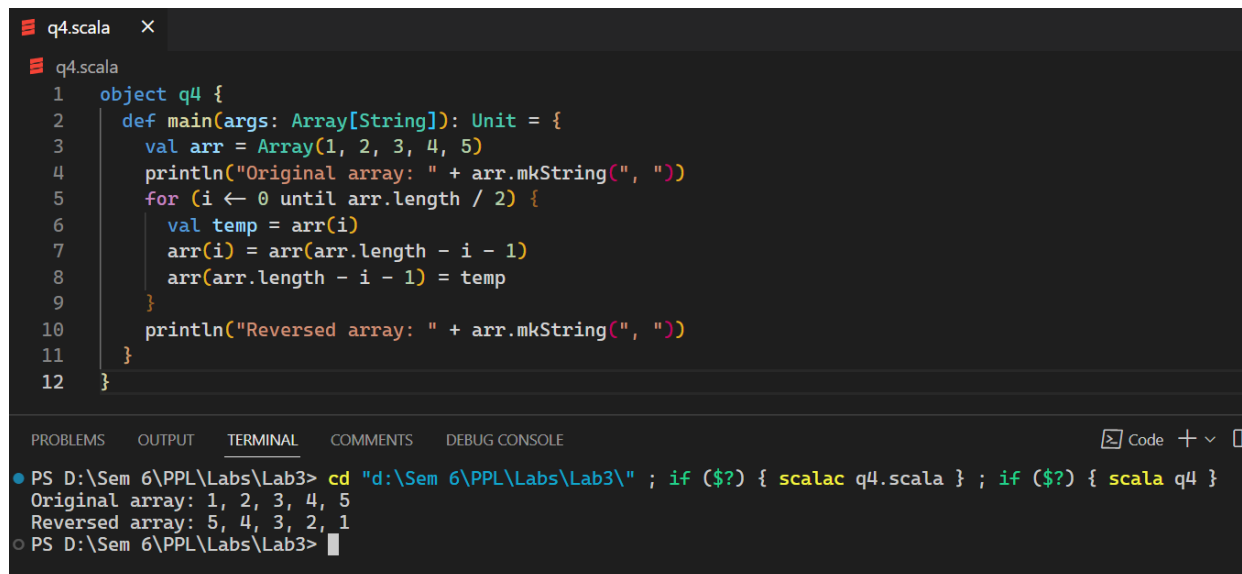
Question 4:

Write a Scala program to reverse an array of integer values.

Code:

```
object q4 {  
  def main(args: Array[String]): Unit = {  
    val arr = Array(1, 2, 3, 4, 5)  
    println("Original array: " + arr.mkString(", "))  
    for (i <- 0 until arr.length / 2) {  
      val temp = arr(i)  
      arr(i) = arr(arr.length - i - 1)  
      arr(arr.length - i - 1) = temp  
    }  
    println("Reversed array: " + arr.mkString(", "))  
  }  
}
```

Output:



The screenshot shows an IDE window with a file named 'q4.scala'. The code is as follows:

```
1 object q4 {  
2   def main(args: Array[String]): Unit = {  
3     val arr = Array(1, 2, 3, 4, 5)  
4     println("Original array: " + arr.mkString(", "))  
5     for (i <- 0 until arr.length / 2) {  
6       val temp = arr(i)  
7       arr(i) = arr(arr.length - i - 1)  
8       arr(arr.length - i - 1) = temp  
9     }  
10    println("Reversed array: " + arr.mkString(", "))  
11  }  
12 }
```

The terminal output shows the execution of the program:

```
PS D:\Sem 6\PPL\Labs\Lab3> cd "d:\Sem 6\PPL\Labs\Lab3\" ; if ($?) { scalac q4.scala } ; if ($?) { scala q4 }  
Original array: 1, 2, 3, 4, 5  
Reversed array: 5, 4, 3, 2, 1  
PS D:\Sem 6\PPL\Labs\Lab3>
```

Lab 3

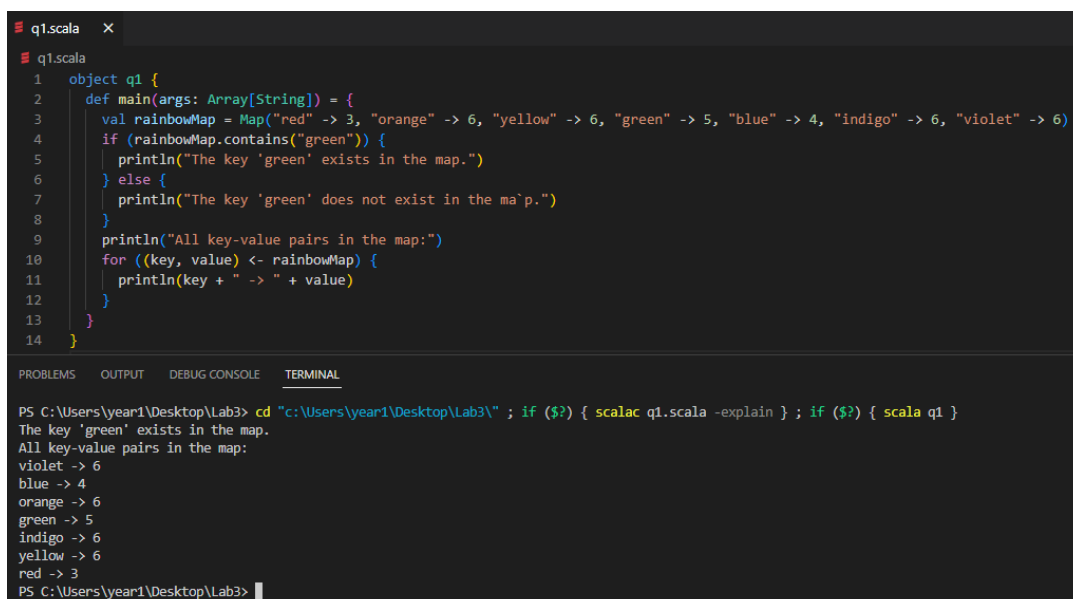
Question 1:

Scala program to implement Maps which contains the colors of rainbow as key and character count as value. Show the use of contains() and display all the key value pairs.

Code:

```
object q1 {  
  def main(args: Array[String]) = {  
    val rainbowMap = Map("red" -> 3, "orange" -> 6, "yellow" -> 6, "green" ->  
5, "blue" -> 4, "indigo" -> 6, "violet" -> 6)  
    if (rainbowMap.contains("green")) {  
      println("The key 'green' exists in the map.")  
    } else {  
      println("The key 'green' does not exist in the ma`p.")  
    }  
    println("All key-value pairs in the map:")  
    for ((key, value) <- rainbowMap) {  
      println(key + " -> " + value)  
    }  
  }  
}
```

Output:



```
q1.scala x  
q1.scala  
1 object q1 {  
2   def main(args: Array[String]) = {  
3     val rainbowMap = Map("red" -> 3, "orange" -> 6, "yellow" -> 6, "green" -> 5, "blue" -> 4, "indigo" -> 6, "violet" -> 6)  
4     if (rainbowMap.contains("green")) {  
5       println("The key 'green' exists in the map.")  
6     } else {  
7       println("The key 'green' does not exist in the ma`p.")  
8     }  
9     println("All key-value pairs in the map:")  
10    for ((key, value) <- rainbowMap) {  
11      println(key + " -> " + value)  
12    }  
13  }  
14 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\year1\Desktop\Lab3> cd "c:\Users\year1\Desktop\Lab3\" ; if ($?) { scalac q1.scala -explain } ; if ($?) { scala q1 }  
The key 'green' exists in the map.  
All key-value pairs in the map:  
violet -> 6  
blue -> 4  
orange -> 6  
green -> 5  
indigo -> 6  
yellow -> 6  
red -> 3  
PS C:\Users\year1\Desktop\Lab3>
```

Question 2:

Write a Scala class that has methods to read the string and it returns whether the entered string is a palindrome or not.

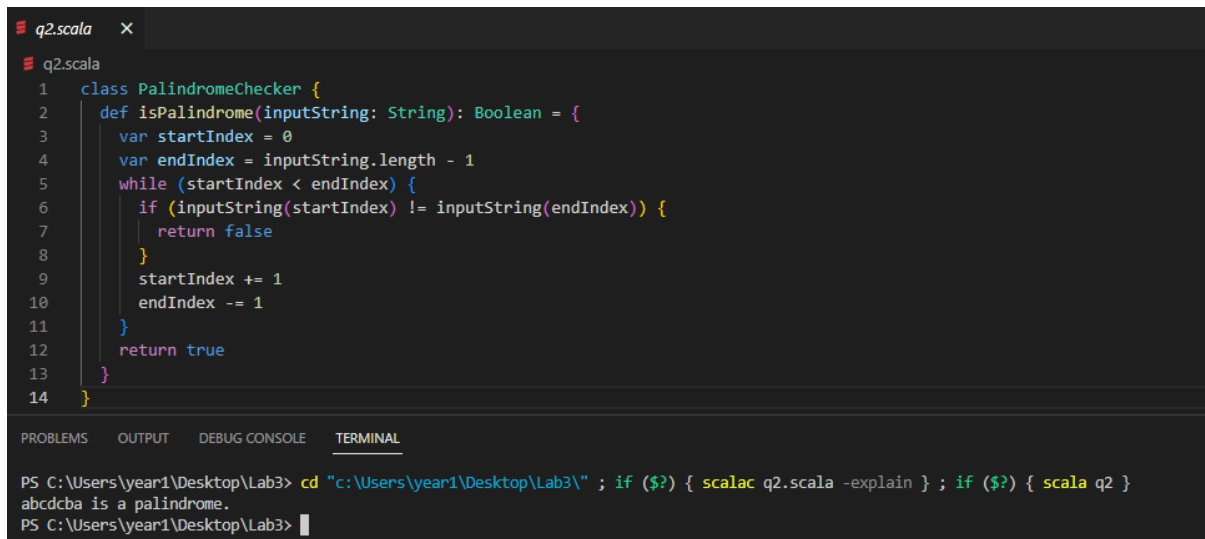
Code:

```
class PalindromeChecker {  
  def isPalindrome(inputString: String): Boolean = {  
    var startIndex = 0  
    var endIndex = inputString.length - 1  
    while (startIndex < endIndex) {  
      if (inputString(startIndex) != inputString(endIndex)) {  
        return false  
      }  
      startIndex += 1  
      endIndex -= 1  
    }  
    return true  
  }  
}  
  
object q2 {  
  def main(args: Array[String]) = {  
    val checker = new PalindromeChecker()  
    val inputString = "abcdcba"  
    val isPalindrome = checker.isPalindrome(inputString)  
    if (isPalindrome) {  
      println(s"$inputString is a palindrome.")  
    }  
    else {  
      println(s"$inputString is not a palindrome.")  
    }  
  }  
}
```

Name - Shubham Garg

Reg No – BL.EN.U4CSE20158

Output:



```
q2.scala x
q2.scala
1 class PalindromeChecker {
2   def isPalindrome(inputString: String): Boolean = {
3     var startIndex = 0
4     var endIndex = inputString.length - 1
5     while (startIndex < endIndex) {
6       if (inputString(startIndex) != inputString(endIndex)) {
7         return false
8       }
9       startIndex += 1
10      endIndex -= 1
11    }
12    return true
13  }
14 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\year1\Desktop\Lab3> cd "c:\Users\year1\Desktop\Lab3\" ; if ($?) { scalac q2.scala -explain } ; if ($?) { scala q2 }
abcdcba is a palindrome.
PS C:\Users\year1\Desktop\Lab3>
```

Question 3:

Write a Scala program to remove duplicates from a list.

Code:

```
object q3 {  
  def main(args: Array[String]): Unit = {  
    val myList = List(1, 2, 2, 3, 4, 4, 5, 5)  
    var uniqueList = List[Int]()  
    for (item <- myList) {  
      if (!uniqueList.contains(item)) {  
        uniqueList = uniqueList :+ item  
      }  
    }  
    println(uniqueList)  
  }  
}
```

Output:



The screenshot shows an IDE window titled 'q3.scala'. The code is the same as in the previous block. Below the code editor, there is a 'TERMINAL' tab. The terminal output shows the command to compile and run the program, followed by the output of the program: 'List(1, 2, 3, 4, 5)'.

```
PS C:\Users\year1\Desktop\Lab3> cd "c:\Users\year1\Desktop\Lab3\" ; if ($?) { scalac q3.scala -explain } ; if ($?) { scala q3 }  
List(1, 2, 3, 4, 5)  
PS C:\Users\year1\Desktop\Lab3>
```

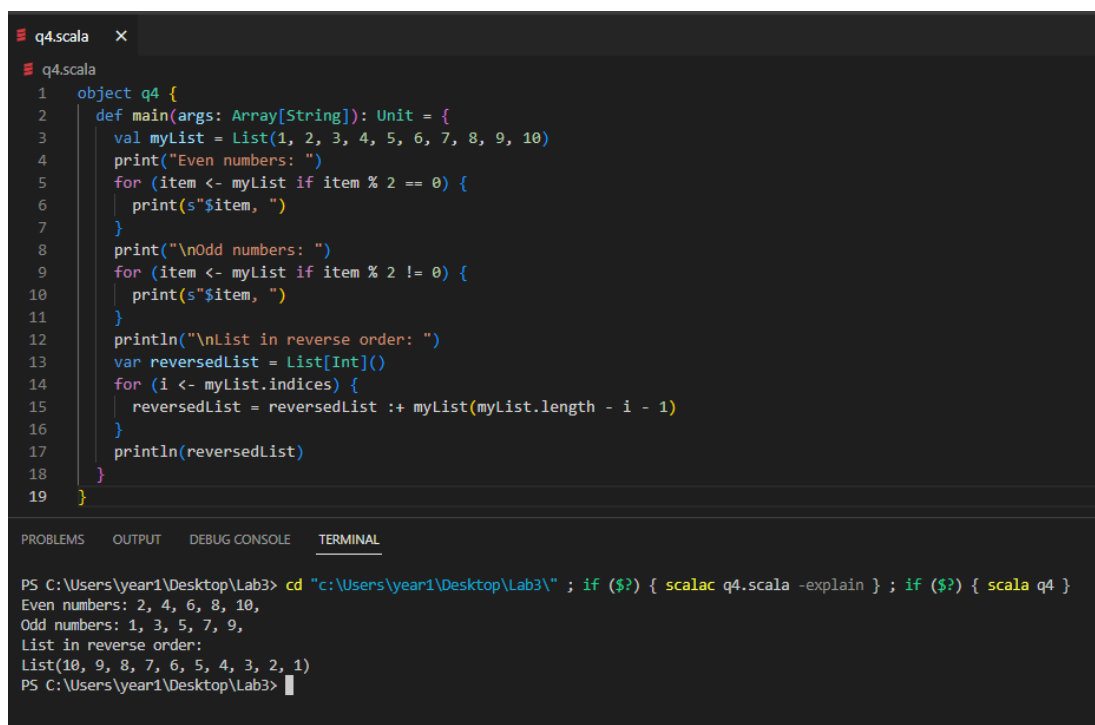
Question 4:

Write a Scala program to remove duplicates from a list.

Code:

```
object q4 {  
  def main(args: Array[String]): Unit = {  
    val myList = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
    print("Even numbers: ")  
    for (item <- myList if item % 2 == 0) {  
      print(s"$item, ")  
    }  
    print("\nOdd numbers: ")  
    for (item <- myList if item % 2 != 0) {  
      print(s"$item, ")  
    }  
    println("\nList in reverse order: ")  
    var reversedList = List[Int]()  
    for (i <- myList.indices) {  
      reversedList = reversedList :+ myList(myList.length - i - 1)  
    }  
    println(reversedList)  
  }  
}
```

Output:



The screenshot shows an IDE window with a file named 'q4.scala'. The code is identical to the one provided in the previous block. Below the code editor, there is a 'TERMINAL' tab showing the execution output. The output is as follows:

```
PS C:\Users\year1\Desktop\Lab3> cd "c:\Users\year1\Desktop\Lab3\" ; if ($?) { scalac q4.scala -explain } ; if ($?) { scala q4 }  
Even numbers: 2, 4, 6, 8, 10,  
Odd numbers: 1, 3, 5, 7, 9,  
List in reverse order:  
List(10, 9, 8, 7, 6, 5, 4, 3, 2, 1)  
PS C:\Users\year1\Desktop\Lab3>
```


Lab 3

Question 1:

Create a class named Rational and implement the basic calculator functions on Rational numbers.

Code:

```
class Rational(val n: Int, val d: Int) {
    require(d != 0, "Denominator must be non-zero")
    override def toString: String = s"$numer/$denom"
    private def gcd(a: Int, b: Int): Int = if (b == 0) a else gcd(b, a % b)
    private val g = gcd(n.abs, d.abs)
    val numer = n / g
    val denom = d / g

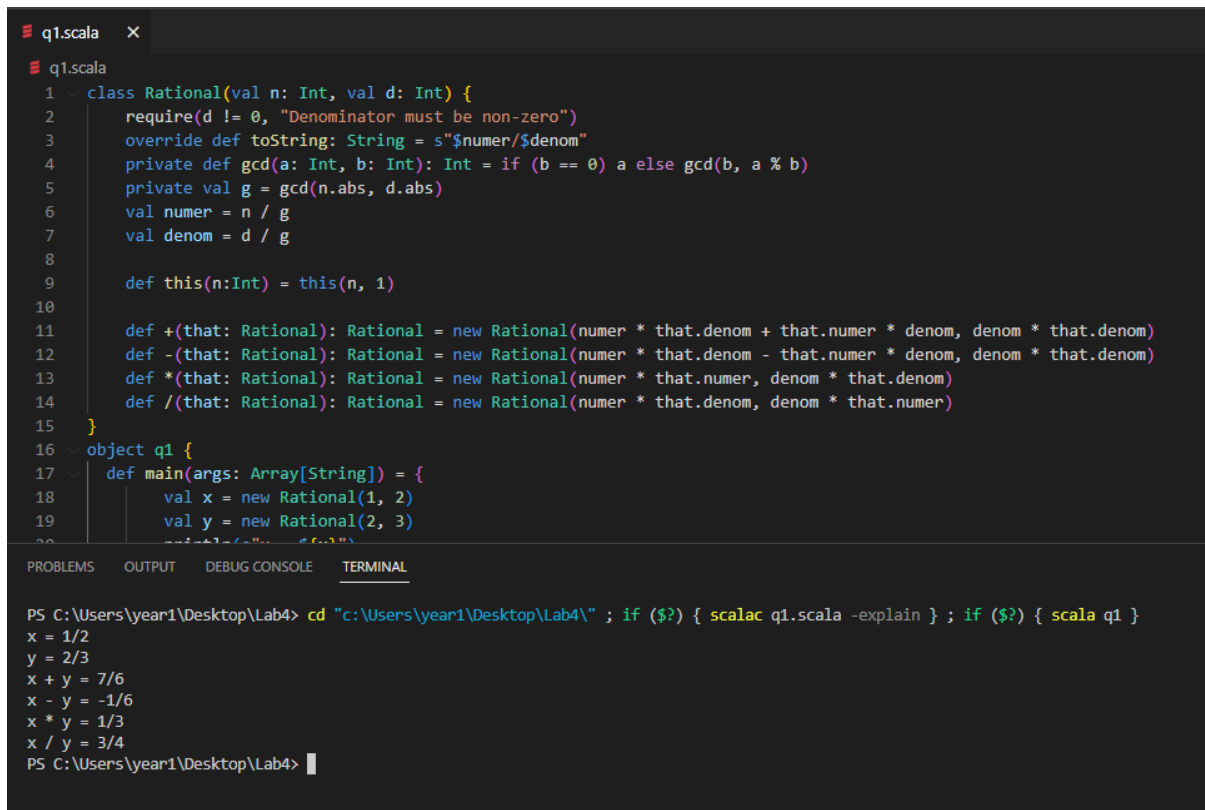
    def this(n: Int) = this(n, 1)

    def +(that: Rational): Rational = new Rational(numer * that.denom +
that.numer * denom, denom * that.denom)
    def -(that: Rational): Rational = new Rational(numer * that.denom -
that.numer * denom, denom * that.denom)
    def *(that: Rational): Rational = new Rational(numer * that.numer, denom *
that.denom)
    def /(that: Rational): Rational = new Rational(numer * that.denom, denom *
that.numer)
}
object q1 {
    def main(args: Array[String]) = {
        val x = new Rational(1, 2)
        val y = new Rational(2, 3)
        println(s"x = ${x}")
        println(s"y = ${y}")
        println(s"x + y = ${x + y}")
        println(s"x - y = ${x - y}")
        println(s"x * y = ${x * y}")
        println(s"x / y = ${x / y}")
    }
}
```

Name - Shubham Garg

Reg No – BL.EN.U4CSE20158

Output:



```
q1.scala x
q1.scala
1 class Rational(val n: Int, val d: Int) {
2   require(d != 0, "Denominator must be non-zero")
3   override def toString: String = s"$numer/$denom"
4   private def gcd(a: Int, b: Int): Int = if (b == 0) a else gcd(b, a % b)
5   private val g = gcd(n.abs, d.abs)
6   val numer = n / g
7   val denom = d / g
8
9   def this(n: Int) = this(n, 1)
10
11   def +(that: Rational): Rational = new Rational(numer * that.denom + that.numer * denom, denom * that.denom)
12   def -(that: Rational): Rational = new Rational(numer * that.denom - that.numer * denom, denom * that.denom)
13   def *(that: Rational): Rational = new Rational(numer * that.numer, denom * that.denom)
14   def /(that: Rational): Rational = new Rational(numer * that.denom, denom * that.numer)
15 }
16 object q1 {
17   def main(args: Array[String]) = {
18     val x = new Rational(1, 2)
19     val y = new Rational(2, 3)
20     println(s"x = $x")
21     println(s"y = $y")
22     println(s"x + y = ${x + y}")
23     println(s"x - y = ${x - y}")
24     println(s"x * y = ${x * y}")
25     println(s"x / y = ${x / y}")
26   }
27 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\year1\Desktop\Lab4> cd "c:\Users\year1\Desktop\Lab4\" ; if ($?) { scalac q1.scala -explain } ; if ($?) { scala q1 }
x = 1/2
y = 2/3
x + y = 7/6
x - y = -1/6
x * y = 1/3
x / y = 3/4
PS C:\Users\year1\Desktop\Lab4>
```