## Assignment #2

Each question and subquestion has a separate Scala code file.

1. Implement the factorial function using *to* and *reduceLeft*, without a loop or recursion.

```
object Question1 extends App {
        println("The␣factorial␣of␣5␣is␣" + f(5))
        println("The␣factorial␣of␣8␣is␣" + f(8))

        def f (n: Int): Int = if (n < 1) 1 else (n to 1 by -1).reduceLeft(_*_)
}
```

2. Write a Scala program to find the prime number from an array of numbers and print them.

```
object Question2 extends App {
        val arr = Array(3,7,4,8,12,13,5,23)
        println("List␣of␣Numbers␣to␣Test:")
        arr.foreach(println)
        println("The␣prime␣numbers␣are:")
        for(i <- arr if isPrime(i)) println(i)

        def isPrime(num:Int): Boolean = {
                if(num <= 1) false
                if(num == 2) true
                List.range(2, num) forall (x => num % x != 0)
        }
}
```

3. Write a Scala code which reads a file and reverse the lines (makes the first line as the last one, and so on). Write the reversed file to a new file named “reversed.txt” at the same location.

```
object Question3 extends App {
        import scala.io.Source
        import java.io._

        reverseLines("alice.txt")

        def reverseLines(f: String){
                println("File␣is␣being␣read.")
                val input = Source.fromFile(f)
                val lines = input.getLines.toArray
                val rev = lines.reverse
                val writer = new PrintWriter(new File("rev.txt"))
                println("File␣is␣being␣written.")
                rev.foreach(writer.write)
                writer.close()
        }
}
```

4. Write a Scala code which reads a file and prints all words with more than 10 characters.

```
object Question4 extends App{
        import scala.io.Source
        import java.io._

        reverseFile("alice.txt", "rev.txt")

        def reverseFile(input: String, output: String){
                println("File␣is␣being␣read.")
                val file = Source.fromFile(input)
                val lines = file.getLines.toArray
                val rev = lines.reverse
                val writer = new PrintWriter(new File(output))
                println("File␣is␣being␣written.")
                rev.foreach(writer.write)
                writer.close()
        }
}
```

5. Write a Scala program to implement QuickSort function. Choose an array of your choice and check the result.

```
object Question5 extends App {
        var x = Array(6,4,8,3,78,46,26,75,13)
        println("Unsorted:")
        x.foreach(println)
        quicksort(x, 0, x.length - 1)
        println("Sorted:")
        x.foreach(println)

        def swap(arr: Array[Int], i : Int, j:Int) {
                val temp = arr(i)
                arr(i) = arr(j)
                arr(j) = temp
        }

        def quicksort(arr: Array[Int], left: Int, right: Int){
                val split = arr((left + right) / 2)
                var i = left
                var j = right

                while(i < j){
                        while(arr(i) < split) i += 1
                        while(arr(j) > split) j -= 1
                        if (i <= j) {
                                swap(arr, i,j)
                                i += 1
                                j -= 1
                        }
                }
                if(left < j) quicksort(arr, left, j)
                if(j < right) quicksort(arr, i,right)
        }
}
```

6. In mathematics, the least common multiple (LCM) of two numbers is the smallest positive integer that can be divided by the two numbers without producing a remainder. LCM can be calculated as follows:

$$LCM(a.b) = \frac{a \cdot b}{GCD(a,b)}$$

where $GCD(a,b)$ is the greatest common divisor of $a$ and $b$, i.e., the largest number that divides both of them without leaving a remainder. Write a Scala program to implement a function to calculate $LCM(a,b)$ using Higher Order Functions.

```scala
object Question6 extends App{

        println("The LCM of 10 and 49 is: " + lcm(10, 40))
        println("The LCM of 65 and 30 is: " + lcm(65, 30))
        println("The LCM of 3 and 5 is: " + lcm(3,5))
        println("The LCM of 6 and 3 is: " + lcm(6, 3))
        println("The LCM of 12 and 48 is: " + lcm(12, 48))

        def gcd(a: Int, b: Int): Int = {
                if(b == 0) a
                else gcd(b, a % b)
        }

        def lcm(a: Int, b: Int): Int = (a * b) / gcd(a, b)
}
```

7. OOP with Scala

   (a) Write a class BankAccount with methods *deposit* and *withdraw*, and read-only property *balance*. Provide customized getter and setter to check the validity of value of *balance*, e,g., *balance* can only initialized with an amount $\geq 0$. Write a main function to test your class.

```scala
object Question7a{
        class BankAccount {
                private var _balance = 0.00
                def this(n: Double){
                        this()
                        if(n >= 0){
                                _balance = n
                        }
                        else{
                                println("This is not tangible money. \n
                                    Current balance is reset to 0.00.")
                        }
                }

                def currentBalance = _balance
                def deposit(d: Double){
                        _balance = _balance + d
                }
                def withdraw(w: Double){
                        if(w <= _balance){
                                _balance = _balance - w
                        }
                        else{
                                println("You don't have this amount of
                                    money.")
                        }
                }
        }
        def main(args: Array[String]){
                println("Instantiate Darshan's account with $100.")
                var darshan = new BankAccount(100)
                println("Current Balance: $" + darshan.currentBalance)
                println("Add $5.")
                darshan.deposit(5)
                println("Current Balance: $" + darshan.currentBalance)
                println("Withdraw $1000.")
                darshan.withdraw(1000)
                println("Current Balance: $" + darshan.currentBalance)
                println("Withdraw $12.95")
                darshan.withdraw(12.95)
                println("Current Balance: $" + darshan.currentBalance)
        }
}
```

(b) Extend your BankAccount class to a CheckingAccount class that charges $1 for every *deposit* and *withdraw*. Write a main function to test your CheckingAccount class.

```
object Question7b{
        class BankAccount {
                private var _balance = 0.00
                def this(n: Double){
                        this()
                        if(n >= 0){
                                _balance = n
                        }
                        else{
                                println("This is not tangible money. \n 
                                    Current balance is reset to 0.00.")
                        }
                }

                def currentBalance = _balance
                def deposit(d: Double){
                        _balance = _balance + d
                }
                def withdraw(w: Double){
                        if(w <= _balance){
                                _balance = _balance - w
                        }
                        else{
                                println("You don't have this amount of 
                                    money.")
                        }
                }
        }
        class CheckingAccount(init: Double) extends BankAccount(init) {
                override def deposit(d: Double){
                        super.deposit(d-1)
                }
                override def withdraw(w: Double){
                        super.withdraw(w+1)
                }
        }
        def main(args: Array[String]){
                println("Instantiate Darshan's account with $5.")
                var darshan = new CheckingAccount(5)
                println("Current Balance: $" + darshan.currentBalance)
                println("Add $5.")
                darshan.deposit(5)
                println("Current Balance: $" + darshan.currentBalance)
                println("withdraw $1000.")
                darshan.withdraw(1000)
                println("Current Balance: $" + darshan.currentBalance)
                println("Withdraw $2.95.")
                darshan.withdraw(2.95)
                println("Current Balance: $" + darshan.currentBalance)
                println("Add $8.50.")
                darshan.deposit(8.50)
                println("Current Balance: $" + darshan.currentBalance)
        }
}
```