Assignment - 15.1

Task1: Create a Scala application to find the GCD of two numbers.

Command used:

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

val gcdIs = gcd(56, 126)
```

```
object Assignment2 {
  println("Welcome to the Scala worksheet")       //> Welcome to the Scala worksheet

  def gcd(a: Int,b: Int): Int = { if(b ==0) a else gcd(b, a%b) }
                                                   //> gcd: (a: Int, b: Int)Int
  val gcdIs = gcd(56, 126)                  //> gcdIs  : Int = 14

}
```

Task 2: Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits.Write a Scala application to find the Nth digit in the sequence.

- Write the function using standard for loop.
- Write the function using recursion

Command used:

To get the Fibonacci series starting from 1 is as below.

```
object Assignment2 {
  println("Welcome to the Scala worksheet")       //> Welcome to the Scala worksheet

  def fibFrom(a: Int, b: Int): Stream[Int] = a #:: fibFrom(b, a + b)
                                                   //> fibFrom: (a: Int, b: Int)Stream[Int]
  val fibs = fibFrom(1, 1).take(10)         //> fibs  : scala.collection.immutable.Stream[Int] = Stream(1, ?)
  val finalSeries = fibs.toList             //> finalSeries  : List[Int] = List(1, 1, 2, 3, 5, 8, 13, 21, 34, 55)
  println(finalSeries)                      //> List(1, 1, 2, 3, 5, 8, 13, 21, 34, 55)
}
```

To get the value of nth digit in the sequence refer the below code.

Fabonacci.scala SquareRoot.scala

```
import scala.annotation.tailrec;

object Fabonacci {
  val fibs:Stream[Int] = 0 #:: 1 #:: (fibs zip fibs.tail).map{ t => t. 1 + t. 2 }

  def main(args: Array[String]) {
    val n: Int = 9
    println(fibs(n))
   }
}
```

Output:

```
Problems  Tasks  Console
<terminated> Fabonacci$ [Scala Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (09-May-2018, 11:50:39 PM)
34
```

Task 3: Find square root of number using Babylonian method.

1. Start with an arbitrary positive start value x (the closer to the root, the better).
1. 2.Initialize y = 1.
2. Do following until desired approximation is achieved.
   a) Get the next approximation for root using average of x and y
   b) Set y = n/x

Command used: The below screenshot of the class is used to get the squareroot in Babylonian method. Run it as a Scala Application.

```
SquareRoot.scala
object SquareRoot {

  def squareRoot(n: BigDecimal): Stream[BigDecimal] =
  {
    def squareRoot(guess: BigDecimal, n: BigDecimal): Stream[BigDecimal] = {
      Stream.cons(guess, squareRoot(0.5 * (guess + n / guess), n))
    }
  squareRoot(1, n)
  }


  def main(args: Array[String]) {
  val n: Int = 36
  println(squareRoot(n))

  val iterations = 10
  println(squareRoot(n)(iterations - 1))

  println(squareRoot(n).take(iterations).toList)

 }
}
```

Output:

I am mentioning the correct set of output here as we can't see the full output in the given screen shot.

```
Stream(1, ?)
6.0000000000000000000000000000000
```

```
List(1, 18.5, 10.222972972972972972972972972972972,
6.872226737643128918740286883049605, 6.055351744849478693088492098465850,
6.000252984119418644112866239495787, 6.000000005333188854487891118197695,
6.000000000000000000237024194436266 4, 6.000000000000000000000000000000000,
6.000000000000000000000000000000000)
```