Task 1

Create a Scala application to find the GCD of two numbers

gcdMain.scala

```scala
package Assignment8

object gcdMain {
  def main(args:Array[String]){
    val a:Int = readLine("Enter the first number: ").toInt
    val b:Int = readLine("Enter the second number: ").toInt
    println("GCD of "+a+" and "+b+" is --> "+ gcd(a, b))
  }

  def gcd(a: Int, b:Int): Int = {
    // Everything divides 0
    if (a == 0 || b == 0)
       return 0
    if (a == b)
        return a
    if (a > b)
        return gcd(a-b, b)
    return gcd(a, b-a)
  }
}
```

Problems Tasks Console

```
<terminated> gcdMain$ [Scala Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Aug 28, 2018,
Enter the first number: 98
Enter the second number: 56
GCD of 98 and 56 is --> 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

gcdMain.scala | FibonacciMain.scala

```scala
package Assignment8

object FibonacciMain {
  def main(args: Array[String]){
    var n1:Int = 1;     var n2:Int = 1;     var n:Int = 0
    var nth:Int = readLine("Enter the required element number to be get from Fibonacci series: ").toInt
    if (nth == 1)
      println("\n"+nth+"th element in the Fibonacci series is "+n1)
    else if (nth == 2)
      println("\n"+nth+"th element in the Fibonacci series is "+n2)
    else {
      print(n1);     print(n2)
      for (c <- 3 to nth){
        n  = n1 + n2;        print(n)
        n1 = n2;          n2 = n
      }
      println("\n\n"+nth+"th element in the Fibonacci series with standard for loop is "+n)
    }
    println("\nNow we will try to achieve same thing using recursive function")
    println(nth+"th element in the Fibonacci series with recursive function is "+ fib(nth))
  }
  def fib( n : Int) : Int = n match {
     case 0 | 1 => n
     case _ => fib( n-1 ) + fib( n-2 )
  }
}
```

Problems | Tasks | Console

```
<terminated> FibonacciMain$ [Scala Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Aug 28, 2018, 4:56:12 PM)
Enter the required element number to be get from Fibonacci series: 8
1123581321

8th element in the Fibonacci series with standard for loop is 21

Now we will try to achieve same thing using recursive function
8th element in the Fibonacci series with recursive function is 21
```

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).
2. Initialize y = 1.
3. 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

squareRoot.scala

```scala
package Assignment8

object squareRoot {
  def squareRoot(n:Float) = {
    var x:Float = n;     var y:Float = 1;
    var e:Double = 0.000001

    while(x-y > e){
      x = (x+y)/2
      y = n/x
    }
    x
  }
  def main(args: Array[String]){
    val n = readLine("Enter the number whose square root needs to be found: ").toFloat
    println("Square root of "+ n +" is "+ squareRoot(n))
  }
}
```

Problems Tasks Console

```
<terminated> squareRoot$ [Scala Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Aug 28, 2018, 5:10:22 PM)
Enter the number whose square root needs to be found: 50
Square root of 50.0 is 7.071068
```

Task 4

Create a calculator to work with rational numbers.
Requirements:

➢ It should provide capability to add, subtract, divide and multiply rational Numbers
➢ Create a method to compute GCD (this will come in handy during operations on rational)

Add option to work with whole numbers which are also rational numbers i.e. (n/1)
➢ achieve the above using auxiliary constructors
➢ enable method overloading to enable each function to work with numbers and rational.

CalObj.scala

```scala
package Assignment8

class Calc(n:Int, d:Int){
  val g = gcd(n.abs,d.abs)
  val num = n/g
  val den = d/g

  def gcd(a:Int, b:Int) : Int = {
    if (a == 0 || b == 0)
       return 0
    if (a == b)
        return a
    if (a > b)
        return gcd(a-b, b)
    return gcd(a, b-a)
  }

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

  def add(r:Calc): Calc = new Calc(num * r.den + r.num * den, den * r.den)
  def add(i:Int): Calc = new Calc(num + i * den, den)

  def sub(r:Calc): Calc = new Calc(num * r.den - r.num * den, den * r.den)
  def sub(i:Int): Calc = new Calc(num - i * den, den)

  def mul(r:Calc): Calc = new Calc(num * r.num, den * r.den)
  def mul(i:Int): Calc = new Calc(num * i, den)

  def div(r:Calc): Calc = new Calc(num * r.den , den * r.num)
  def div(i:Int): Calc = new Calc(num, den * i)

  def display() : Unit = println(num + "/" + den)
}
```

```scala
object CalObj {
  def main(args: Array[String]){
    val n = new Calc(10,20)

    val addVar = n add 5
    addVar.display()

    val subVar = n sub new Calc(5,30)
    subVar.display()

    val mulVar = n mul new Calc(5,4)
    mulVar.display()

    val divVar = n div 50
    divVar.display()
  }
}
```

Problems Tasks Console

```
<terminated> CalObj$ [Scala Application] C:\Program Files\Java\jre1.8.0_181\b
11/2
1/3
5/8
1/100
```