Task 1

Create a Scala application to find the GCD of two numbers

```
scala> def gcd(num1:Int, num2: Int):Int = {
     | if(num2 == 0) num1 else gcd(num2, num1%num2)
     | }
gcd: (num1: Int, num2: Int)Int

scala> print(gcd(25,15))
5
scala>
```

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

```
scala> def fibUsingLoop(n: Int): Int = {
     |
     |     var first = 0
     |     var second = 1
     |     var count = 0
     |
     |     while(count < n){
     |       val sum = first + second
     |       first = second
     |       second = sum
     |       count = count + 1
     |     }
     |
     |     return first
     | }
fibUsingLoop: (n: Int)Int

scala> fib
fib2   fibUsingLoop

scala> fibUsingLoop(5)
res14: Int = 5

scala> fibUsingLoop(4)
res15: Int = 3

scala> fibUsingLoop(6)
res16: Int = 8
```

```
scala> def fibUsingRecursive(n: Int): Int = {
     |     def fib_tail(n: Int, a: Int, b: Int): Int = n match {
     |       case 0 => a
     |       case _ => fib_tail(n - 1, b, a + b)
     |     }
     |     return fib_tail(n, 0 , 1)
     | }
fibUsingRecursive: (n: Int)Int

scala> print(fibUsingRecursive(6))
8
scala>
```

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

```
scala> def squareRoot(n: Float): Float = {
     |     var x: Float = n
     |     var y: Float = 1
     |     val e: Double = 0.000001
     |     while (x - y > e) {
     |       x = (x + y) / 2
     |       y = n / x
     |     }
     |     return x
     | }
squareRoot: (n: Float)Float

scala> squareRoot(49)
res1: Float = 7.0

scala> squareRoot(50)
res2: Float = 7.071068
```