Assignment SCALA 2

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> println(gcd(14,21))
7
```

Explanation:

gcd(14, 21) → if (21 == 0) 14 else gcd(21, 14 % 21)

→ if (false) 14 else gcd(21, 14 % 21)

→ gcd(21, 14 % 21) → gcd(21, 14)

→ if (14 == 0) 21 else gcd(14, 21 % 14)

→ → gcd(14, 21 % 14)

→ gcd(14, 7)

→ if (7 == 0) 14 else gcd(7, 14 % 7)

→ → gcd(7, 14 % 7)

→ gcd(7, 0)

→ if (0 == 0) 7 else gcd(0, 7 % 0)

→ → 7---result

Assignment SCALA 2

Task 2: Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits.

a) Write the function using standard for loop

```
scala> def fibUsingLoop(n : Int) : Int = {
     | var num1=0
     | var num2=1
     | var count=0
     | while(count<n) {
     | val sum = num1 +num2
     | num1 = num2
     | num2 = sum
     | count = count+1
     | }
     | return num1
     | }
fibUsingLoop: (n: Int)Int

scala> println(fibUsingLoop(6))
8
```

Expalanation:

0 1 1 2 3 5 8----→if we consider output from 0 as first number then 6th number in series would be 8.

b)Write the function using recursion

Logic/CODE:

```
scala> def fibRecursive( 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)
     | }
fibRecursive: (n: Int)Int
```

Result:

```
scala> println(fibRecursive(6))
8
```

Assignment SCALA 2

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

LOGIC:

```
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
```

RESULT:

```
scala> println(squareRoot(25))
5.0
```