ASSIGNMENT

Task1:

Command used:

- ➔ def gcd(a: Int, b: Int): Int = if(b==0) a else gcd(b, a%b)
- ➔ gcd(14, 21)

```
acadgild@localhost:~

scala> def gcd(a: Int, b: Int): Int = if (b == 0) a else gcd(b, a % b)
gcd: (a: Int, b: Int)Int

scala> gcd(14, 21)
res3: Int = 7

scala>
```

Task2

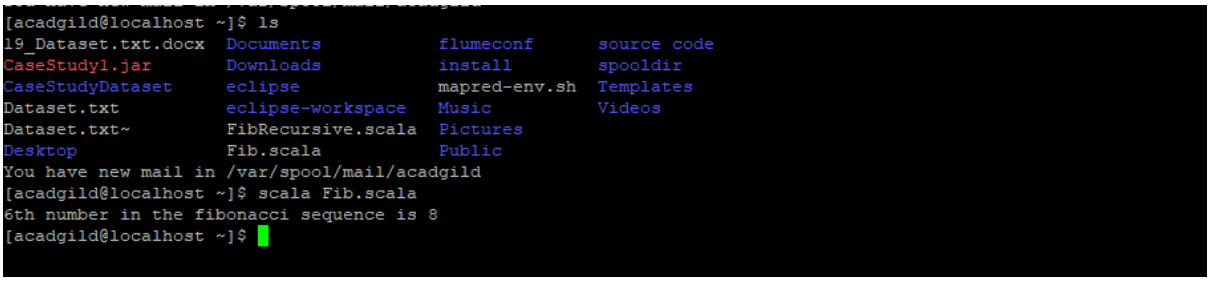
a. Fibonacci series:

```
acadgild@localhost:~
scala> print(pervious1)
0
scala> print(previous2)
0
scala> counter
res6: Int = 10

scala> current
res7: Int = 0

scala> var previous1 = 1
previous1: Int = 1

scala> var previous2 =0
previous2: Int = 0

scala> var counter = 0
counter: Int = 0

scala> while(counter <10){
     | counter = counter+1
     | current = previous1 + previous2
     | previous2 = previous1
     | previous1 = current
     | print(current)}
123581321345589
scala>

scala>

scala>

scala>

scala>
```

b. Write function using standard for loop to find nth digit in the sequence

Sol:

```
object fibonacciSeriesUsingRecursion {
def main(args: Array[String]) {
 val input = 6
 val result = getFibonacci(input)                                    // method
invocation
 println(input + "th number in the fibonacci sequence is " + result)  // print result
 }
 def getFibonacci(number: Int): Int = {
 if(number <= 1)                                                     // check for base
value condition
 number
 else
getFibonacci(number - 1) + getFibonacci(number - 2)                  // invoke
the function
                                                                     // itself recursively taking two
preceding values as input and calculatesum of them
 }
}
```



c. Write function using recursive function to find nth digit in the sequence

Sol:

```
object fibonacciSeriesUsingForLoop {
def main(args: Array[String]) {
 val input = 8
 val result = getNthNumberInFibonacciSequence(input)      // method invocation
 println(input + "th number in the fibonacci sequence is " + result) // print result
 }
def getNthNumberInFibonacciSequence(number: Int): Int = {
 if(number <= 1)                  // check for base value condition
 number
 else {
 val fibonacciArr = new Array[Int](number+1)          //define an array of size number+1
 fibonacciArr(0) = 0              // first value in the Fibonacci series
 fibonacciArr(1) = 1              //second value in the Fibonacci series
 for(i <- 2 to number)            // loop until the given number
 fibonacciArr(i) = fibonacciArr(i-1) + fibonacciArr(i-2)          // store sum of
 fibonacciArr(number)             // two preceding values and return result
 }                                // from nth element of the array
 }
}
```

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ ls
19_Dataset.txt.docx  Documents           flumeconf      source code
CaseStudy1.jar       Downloads           install        spooldir
CaseStudyDataset     eclipse             mapred-env.sh  Templates
Dataset.txt          eclipse-workspace   Music          Videos
Dataset.txt~         FibRecursive.scala  Pictures
Desktop              Fib.scala           Public
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ scala Fib.scala
6th number in the fibonacci sequence is 8
[acadgild@localhost ~]$ scala FibRecursive.scala
8th number in the fibonacci sequence is 21
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
[acadgild@localhost ~]$
```

Task3 : Find square root of number using Babylonian metho

Sol:

```
acadgild@localhost:~
       ^

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

scala> squareRoot(2)
res10: Stream[BigDecimal] = Stream(1, ?)

scala> val iterations =5
iterations: Int = 5

scala> squareRoot(2)(iterations - 1)
res11: BigDecimal = 1.41421356237468991062629557889013 5

scala> squareRoot(2).take(iterations).toList
res12: List[BigDecimal] = List(1, 1.5, 1.41666666666666666666666
6666666666, 1.41421568627450980392156862745098 0, 1.4142135623746
8991062629557889013 5)

scala>

scala>

scala>

scala>

scala>

scala>
```