Scala Basics 2

Task 1

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

In below code, we have converted two input variables into var type so that their values can be changed. Then we have handled negative numbers by negating number if it is less than 0.
Then we have created a new method **find_GCD** and in this method, we have checked if one number is greater than another then we have subtracted small number from big number and reassigned the result to big number variable. We have repeated this step unless and until these two numbers both becomes same by using **while** loop.
We have called this **find_GCD** method in **main** method and displayed the result by using **println** command.
We have taken two numbers 60 and 24 to find GCD. Then we have GCD result as 12.

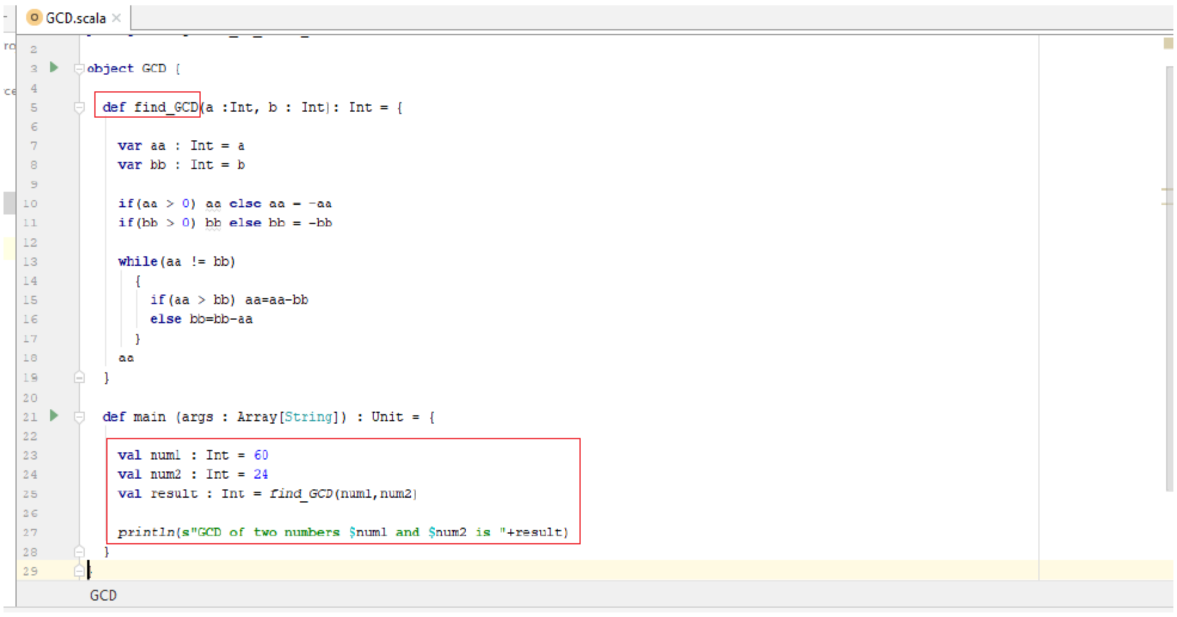
Scala Program :

```
package Assignment_15_Scala_2

object GCD {

       def find_GCD(a :Int, b : Int): Int = {

              var aa : Int = a
              var bb : Int = b

              if(aa > 0) aa else aa = -aa
              if(bb > 0) bb else bb = -bb

              while(aa != bb)
                     {
                            if(aa > bb) aa=aa-bb
                            else bb=bb-aa
                     }
                     aa
                }

              def main (args : Array[String]) : Unit = {

              val num1 : Int = 60
              val num2 : Int = 24

              val result : Int = find_GCD(num1,num2)

              println(s"GCD of two numbers $num1 and $num2 is "+result)
              }
             }
```

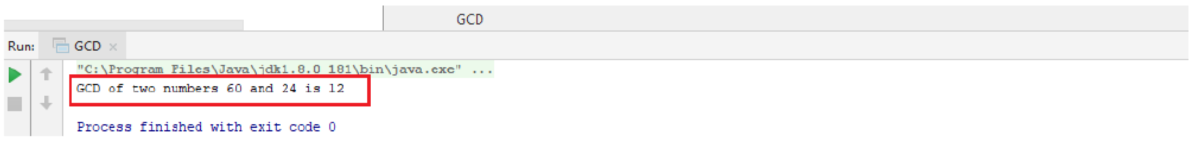
Program Output :

```
GCD of two numbers 60 and 24 is 12
```

Scala Basics 2



```
object GCD {

  def find_GCD(a :Int, b : Int): Int = {

    var aa : Int = a
    var bb : Int = b

    if(aa > 0) aa else aa = -aa
    if(bb > 0) bb else bb = -bb

    while(aa != bb)
      {
        if(aa > bb) aa=aa-bb
        else bb=bb-aa
      }
    aa
  }

  def main (args : Array[String]) : Unit = {

    val num1 : Int = 60
    val num2 : Int = 24
    val result : Int = find_GCD(num1,num2)

    println(s"GCD of two numbers $num1 and $num2 is "+result)
  }
}
```



```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
GCD of two numbers 60 and 24 is 12

Process finished with exit code 0
```

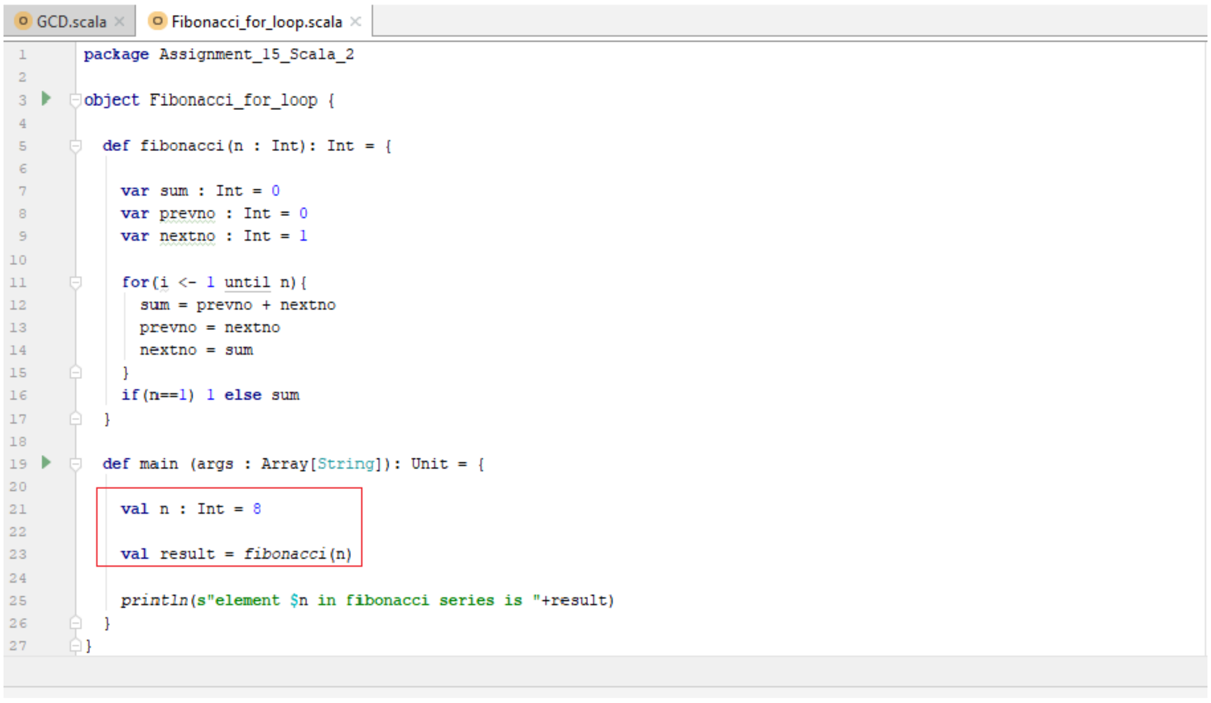
Task 2
Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits.
Write a Scala application to find the Nth digit in the sequence.
➢ Write the function using standard for loop
➢ Write the function using recursion
Here we have Fibonacci series as **1 1 2 3 5 8 13 21 34** and so on.
➢ Write the function using standard for loop :
In below code, we have created a new method **fibonacci** and in this method, we have used **FOR** loop and used three variables **sum, prevno** and **nextno** with values as 0, 0 and 1 respectively.
Then we have swapped these values and taken the **sum** value from this method.
We have called this **fibonacci** method in **main** method and displayed the result by using **println** command.
We have found out 8th element in Fibonacci series in this program and got the result as 21.
As in Fibonacci series: 1 1 2 3 5 8 13 **21** 34 we could verify that 21 is 8th element.

__Scala Program__ :

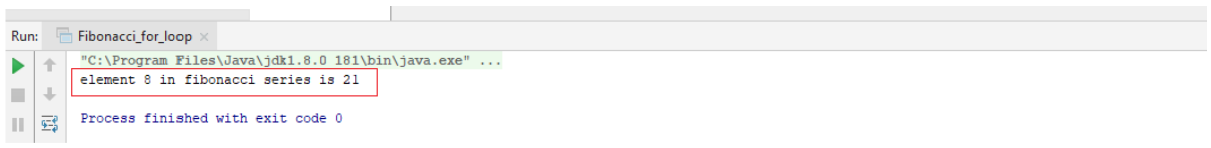
```
object fibonacci_for_loop {

def fibonacci(n : Int): Int = {

       var sum : Int = 0
       var prevno : Int = 0
       var nextno : Int = 1

for(i <- 1 until n){
        sum = prevno + nextno
        prevno = nextno
        nextno = sum
       }
       if(n==1) 1 else sum
       }

       def main (args : Array[String]): Unit = {

       val n : Int = 8
       val result = fibonacci(n)
       println(s"element $n in fibonacci series is "+result) } }
```

Program Output :

element 8 in fibonacci series is 21



```
GCD.scala    Fibonacci_for_loop.scala

package Assignment_15_Scala_2

object Fibonacci_for_loop {

  def fibonacci(n : Int): Int = {

    var sum : Int = 0
    var prevno : Int = 0
    var nextno : Int = 1

    for(i <- 1 until n){
      sum = prevno + nextno
      prevno = nextno
      nextno = sum
    }
    if(n==1) 1 else sum
  }

  def main (args : Array[String]): Unit = {

    val n : Int = 8

    val result = fibonacci(n)

    println(s"element $n in fibonacci series is "+result)
  }
}
```



```
Run: Fibonacci_for_loop
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
element 8 in fibonacci series is 21

Process finished with exit code 0
```

Write the function using recursion

In below code, we have created **fibonacci** method. As we know that first two elements in Fibonacci series are 1. Hence in this method, it will return 1 for 1st and 2nd element. These values are being used in recursion.

We have called this **fibonacci** method in **main** method and displayed the result by using **println** command.

We have found out 8th element in Fibonacci series in this program and got the result as 21.

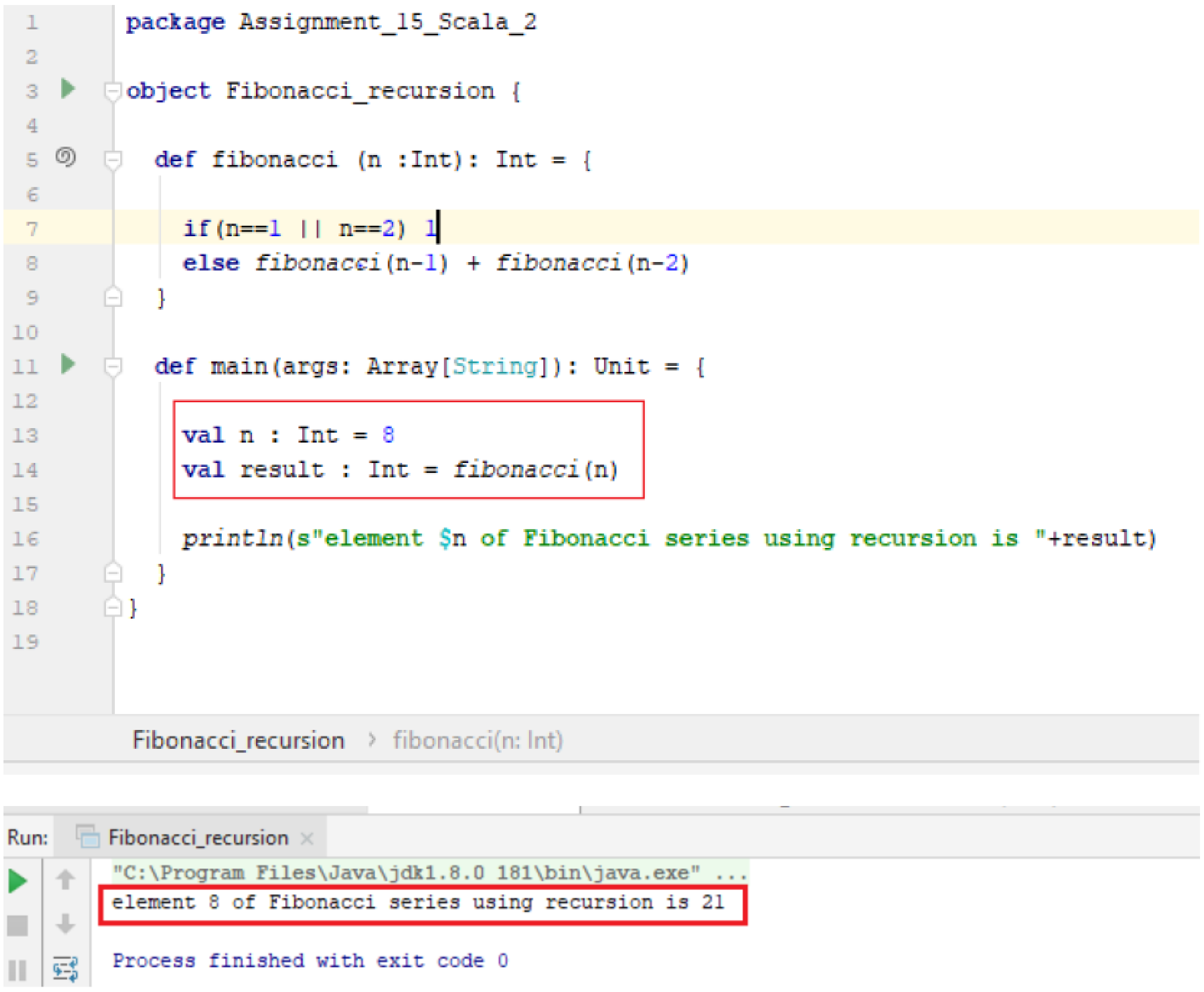
As in Fibonacci series: 1 1 2 3 5 8 13 **21** 34 we could verify that 21 is 8th element.

Scala Program:

Scala Basics 2

```scala
package Assignment_15_Scala_2

object Fibonacci_recursion {

  def fibonacci  (n :Int): Int = {

    if(n==1 || n==2) 1
    else fibonacci(n-1) + fibonacci(n-2)
  }

  def main(args: Array[String]): Unit = {

    val n : Int = 8
    val result : Int = fibonacci(n)

    println(s"element $n of Fibonacci series using recursion is "+result)
  }
}
```

Program Output :
element 8 of Fibonacci series using recursion is 21



```scala
package Assignment_15_Scala_2

object Fibonacci_recursion {

  def fibonacci  (n :Int): Int = {

    if(n==1 || n==2) 1
    else fibonacci(n-1) + fibonacci(n-2)
  }

  def main(args: Array[String]): Unit = {

    val n : Int = 8
    val result : Int = fibonacci(n)

    println(s"element $n of Fibonacci series using recursion is "+result)
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
element 8 of Fibonacci series using recursion is 21

Process finished with exit code 0
```

Scala Basics 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

In below code, we have created a new method **squareRoot** and in this method, we have used **WHILE** loop and used three Float variables **y,x** and **e** with their values as 1, n and 0.001 respectively.

Here we have used **Babylonian method** to find out Square Root.

Variable **e** is used for the accuracy level. Lesser the value of **e**, more is the accuracy.

We have called this **squareRoot** method in **main** method and displayed the result by using **println** command.

We have found out Square Root of **16** in this program and got the result as **4.0000005** which is approximate to exact square root i.e. **4**.

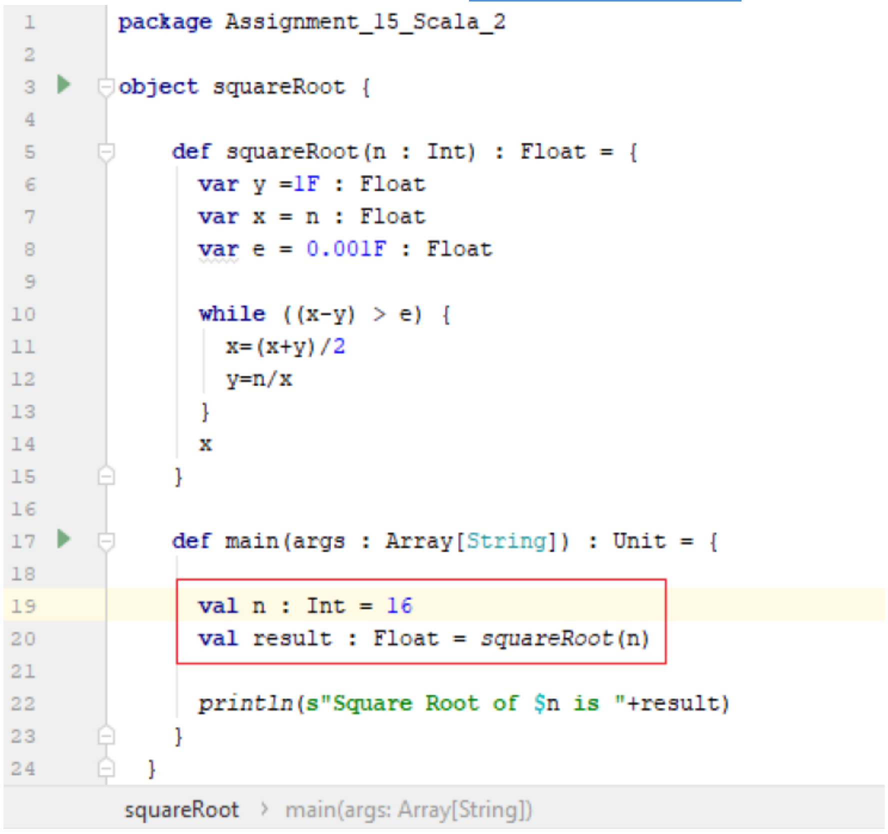
Scala Program :

```
package Assignment_15_Scala_2
object squareRoot {

def squareRoot(n : Int) : Float = {

       var y =1F : Float
       var x = n : Float
       var e = 0.001F : Float

while ((x-y) > e) {
       x=(x+y)/2 y=n/x } x }

def main(args : Array[String]) : Unit = {
       val n : Int = 16 val result : Float = squareRoot(n)

       println(s"Square Root of $n is "+result) } }
```

Output :

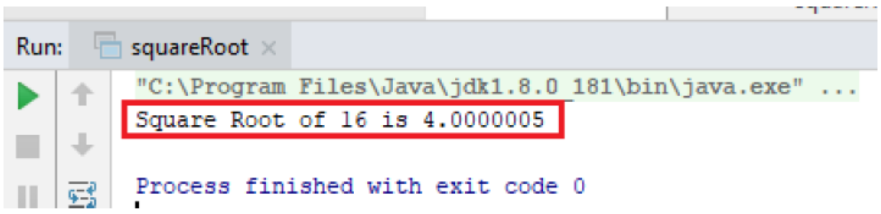
Square Root of 16 is 4.0000005

Scala Basics 2



```
package Assignment_15_Scala_2

object squareRoot {

    def squareRoot(n : Int) : Float = {
      var y =1F : Float
      var x = n : Float
      var e = 0.001F : Float

      while ((x-y) > e) {
        x=(x+y)/2
        y=n/x
      }
      x
    }

    def main(args : Array[String]) : Unit = {

      val n : Int = 16
      val result : Float = squareRoot(n)

      println(s"Square Root of $n is "+result)
    }
  }
```

squareRoot › main(args: Array[String])



Run: squareRoot

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Square Root of 16 is 4.0000005

Process finished with exit code 0
```