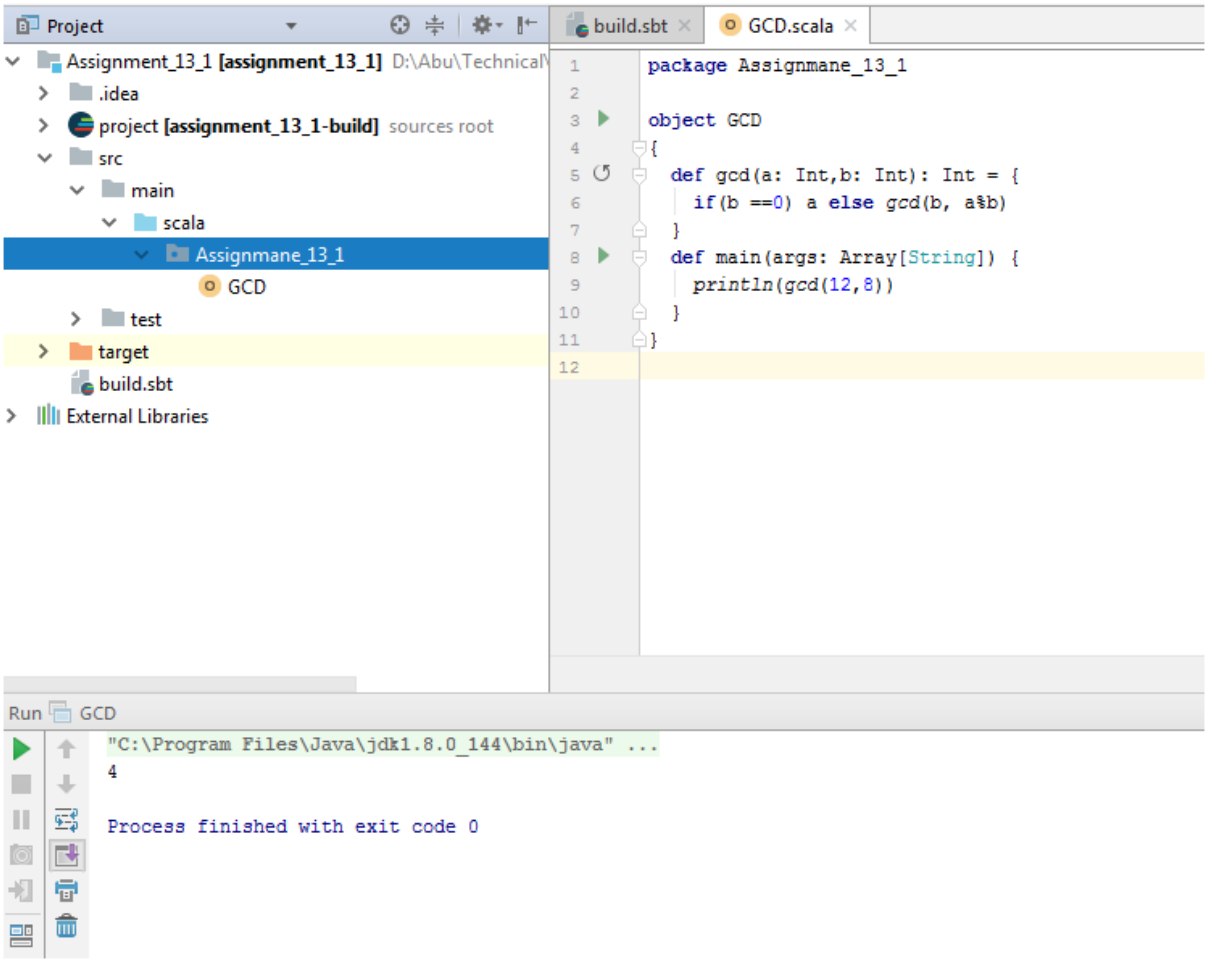
Task 1

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

```
package Assignmane_13_1                    //package which we created

object GCD   //a new object GCD is created
{
  def gcd(a: Int,b: Int): Int = {          // declaring a function gcd
    if(b == 0) a else gcd(b, a%b)          2 integer variables a,b
  }
  def main(args: Array[String])            // Our main function takes in a
named
{                                    parameter args which is an Array of String.
    println(gcd(12,8))                     //  print the result
  }
}
```



Task 2

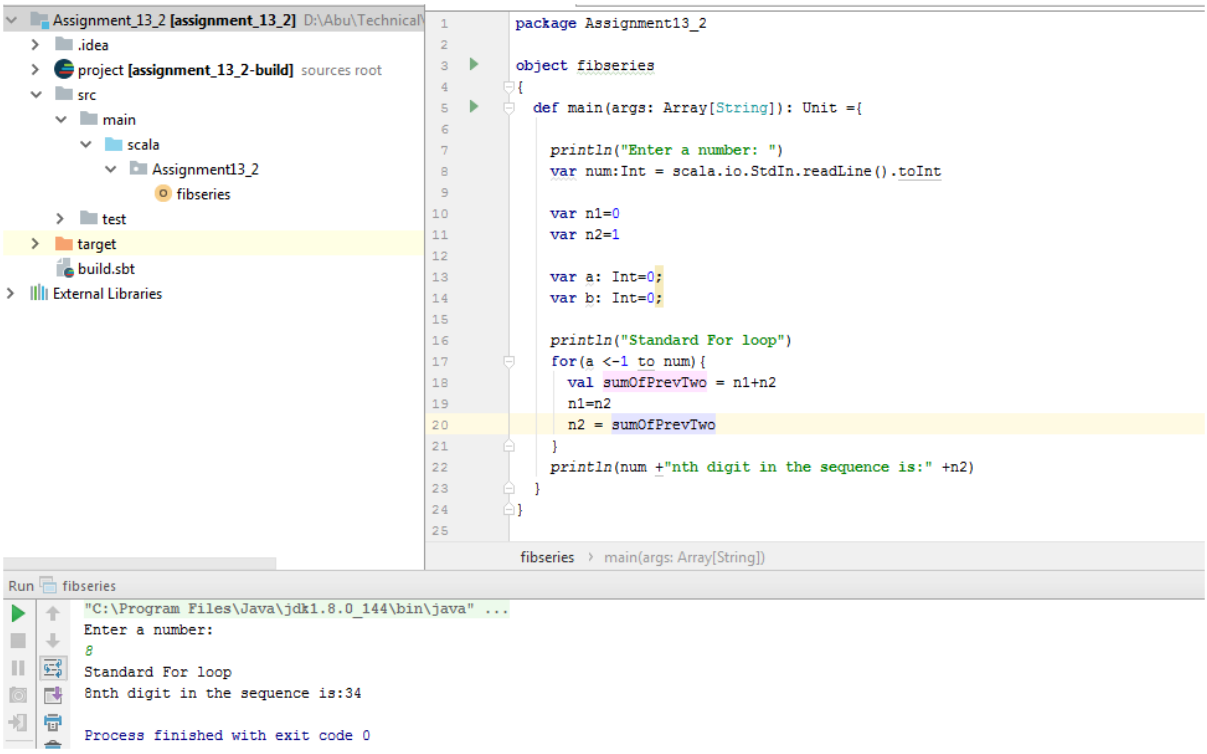
Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits.
Write a Scala application to find the Nth digit in the sequence.

➢ Write the function using standard for loop

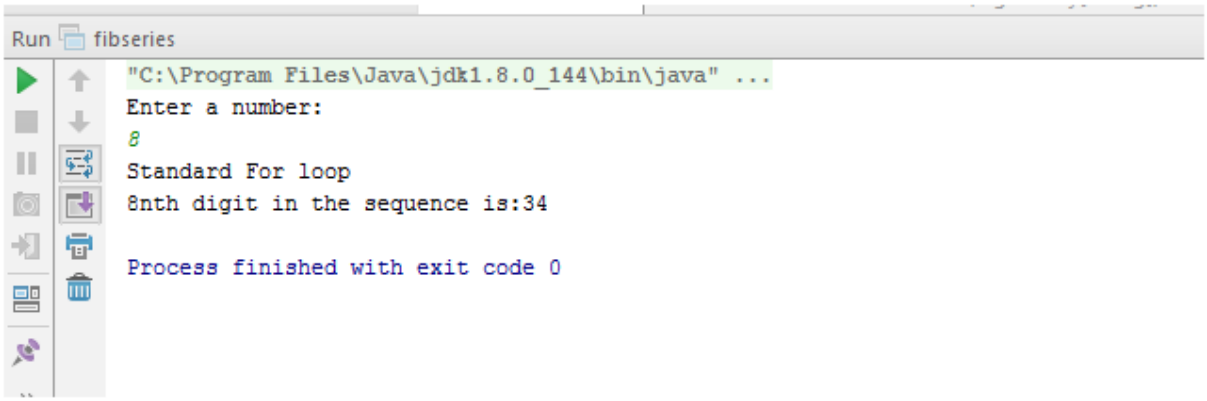
➢ Write the function using recursion

Write the function using standard for loop

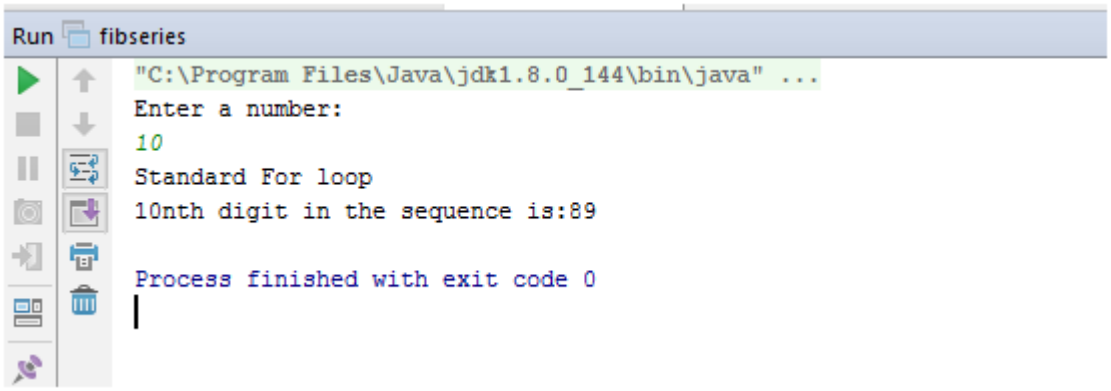
```
package Assignment13_2

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

    println("Enter a number: ")
    var num:Int = scala.io.StdIn.readLine().toInt

    var n1=0
    var n2=1

    var a: Int=0;
    var b: Int=0;

    println("Standard For loop")
    for(a <-1 to num){
      val sumOfPrevTwo = n1+n2
      n1=n2
      n2 = sumOfPrevTwo
    }
    println(num +"nth digit in the sequence is:" +n2)
  }
}
```



When we provide number 8 as input, the 8th digit in the Fibonacci sequence is 34.

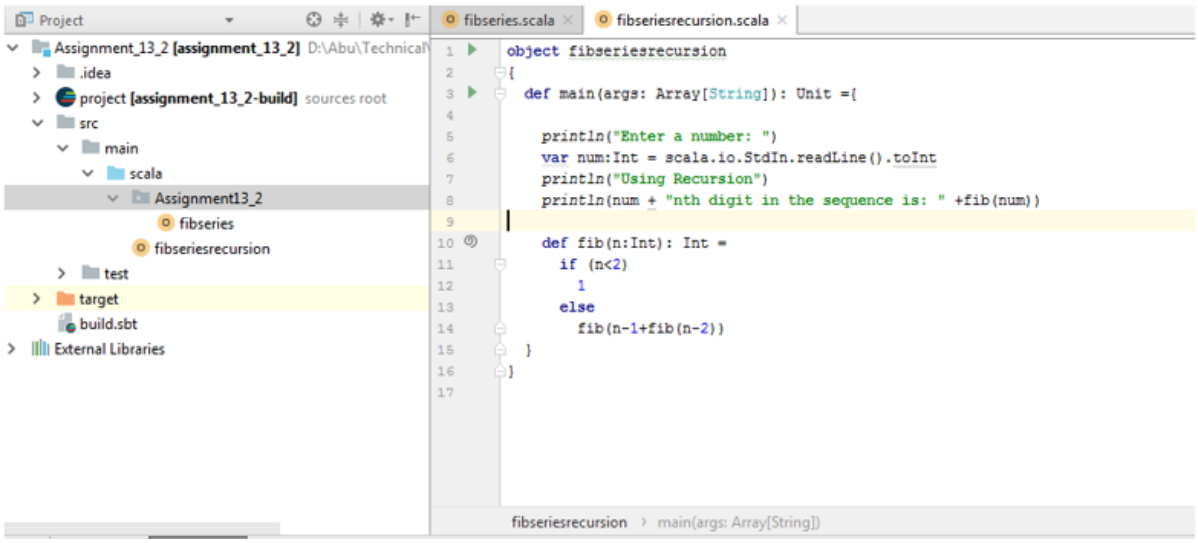


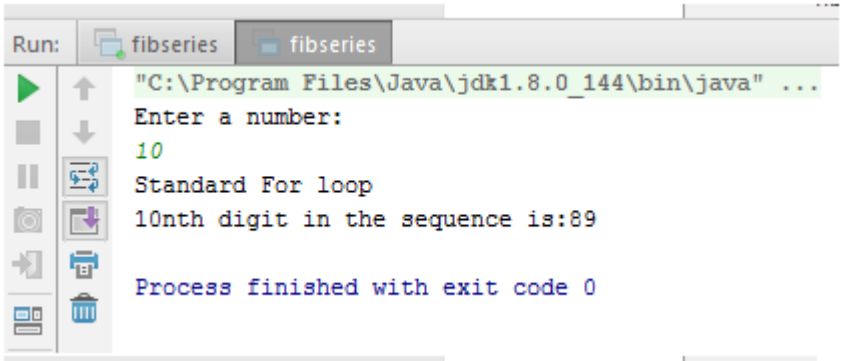
If we give the input as 10, the 10th digit of Fibonacci sequence is 89

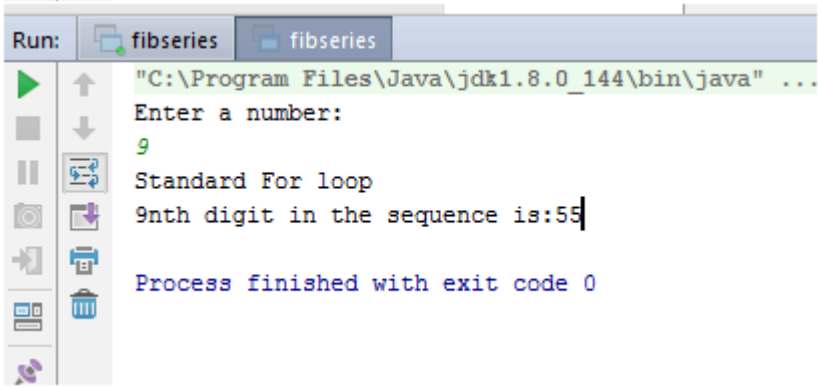


Write the function using recursion

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

    println("Enter a number: ")
    var num:Int = scala.io.StdIn.readLine().toInt
    println("Using Recursion")
    println(num + "nth digit in the sequence is: " +fib(num))

  def fib(n:Int): Int =
    if (n<2)
      1
    else
      fib(n-1+fib(n-2))
  }
}
```


Project
Assignment_13_2 [assignment_13_2] D:\Abu\Technical\
.idea
project [assignment_13_2-build] sources root
src
main
scala
Assignment13_2
fibseries
fibseriesrecursion
test
target
build.sbt
External Libraries
fibseries.scala
fibseriesrecursion.scala
object fibseriesrecursion
{
def main(args: Array[String]): Unit ={
println("Enter a number: ")
var num:Int = scala.io.StdIn.readLine().toInt
println("Using Recursion")
println(num + "nth digit in the sequence is: " +fib(num))
def fib(n:Int): Int =
if (n<2)
1
else
fib(n-1+fib(n-2))
}
}
fibseriesrecursion › main(args: Array[String])


Run:
fibseries
fibseries
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
10
Standard For loop
10nth digit in the sequence is:89
Process finished with exit code 0

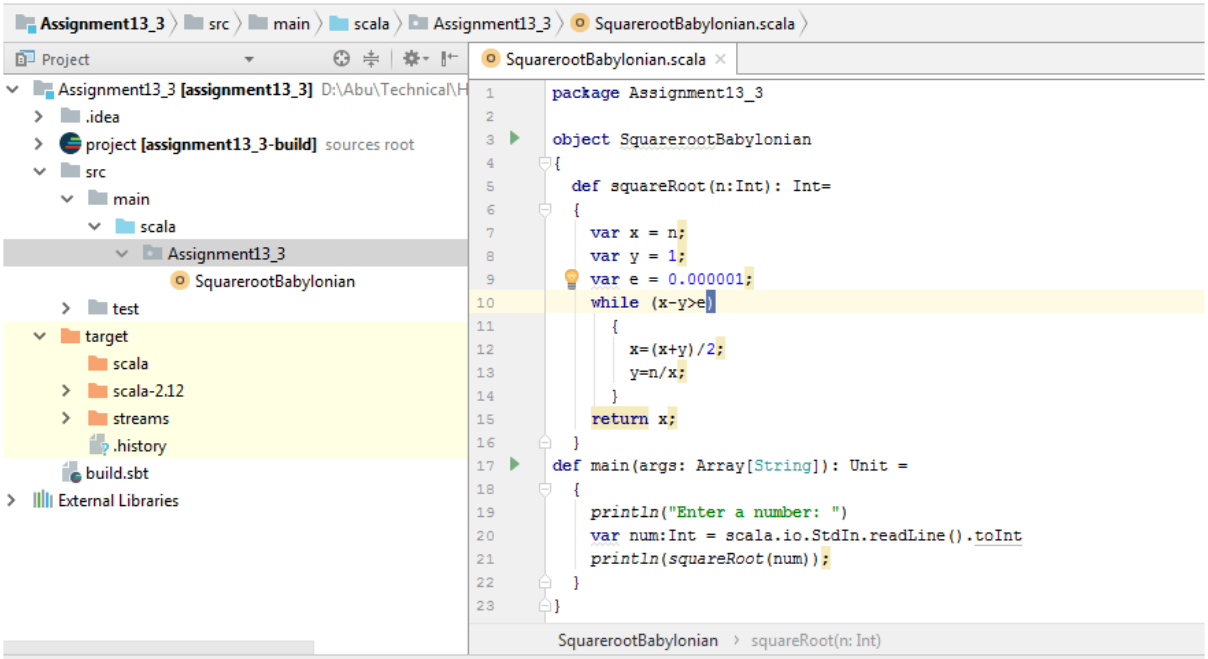

Run:
fibseries
fibseries
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
9
Standard For loop
9nth digit in the sequence is:55
Process finished with exit code 0

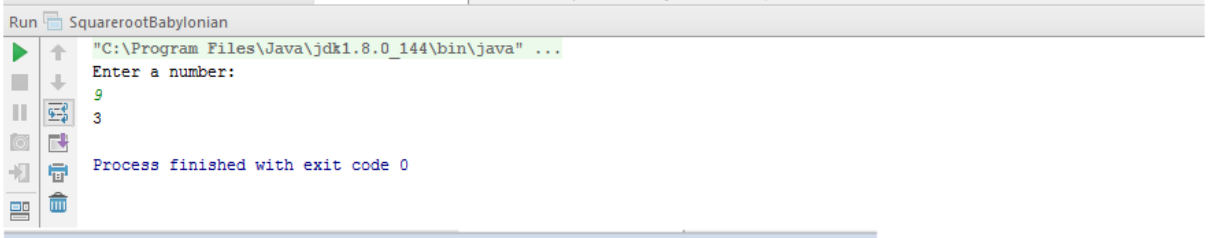
Task 3

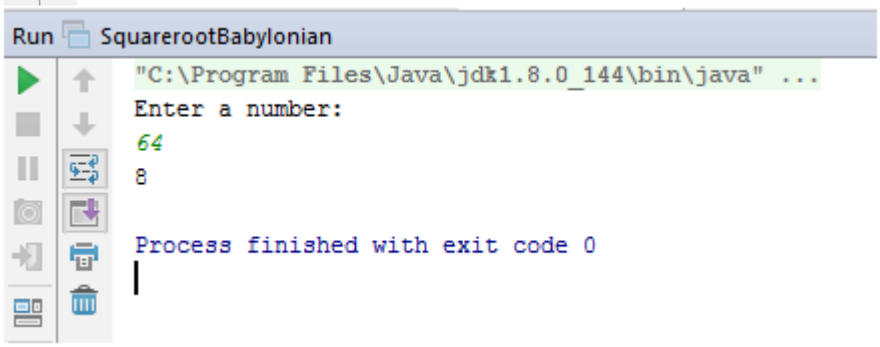
Find square root of number using Babylonian method.

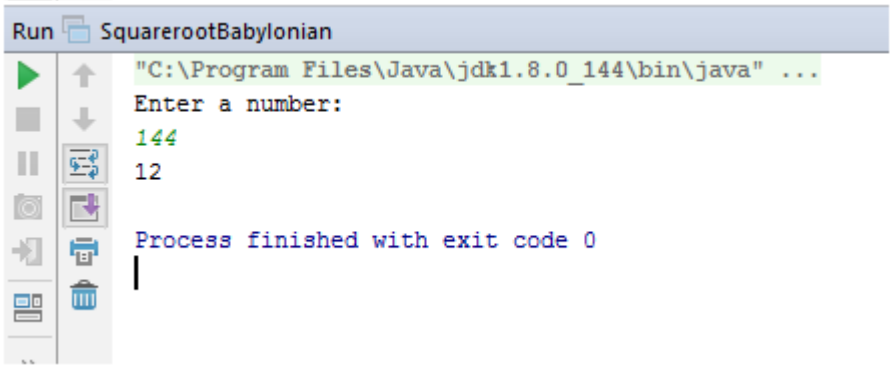
1. Start with an arbitrary positive start value x (the closer to the root, the better).
2. Initialize y = 1.
3. Do following until desired approximation is achieved.
    a) Get the next approximation for root using average of x and y
    b) Set y = n/x

```
package Assignment13_3

object SquarerootBabylonian
{
  def squareRoot(n:Int): Int=
  {
    var x = n;
    var y = 1;
    var e = 0.000001;
    while (x-y>e)
      {
        x=(x+y)/2;
        y=n/x;
      }
    return x;
  }
def main(args: Array[String]): Unit =
  {
    println("Enter a number: ")
    var num:Int = scala.io.StdIn.readLine().toInt
    println(squareRoot(num));
  }
}
```


Assignment13_3 src main scala Assignment13_3 SquarerootBabylonian.scala
Project
Assignment13_3 [assignment13_3] D:\Abu\Technical\H
.idea
project [assignment13_3-build] sources root
src
main
scala
Assignment13_3
SquarerootBabylonian
test
target
scala
scala-2.12
streams
.history
build.sbt
External Libraries
SquarerootBabylonian.scala
1 package Assignment13_3
2
3 object SquarerootBabylonian
4 {
5 def squareRoot(n:Int): Int=
6 {
7 var x = n;
8 var y = 1;
9 var e = 0.000001;
10 while (x-y>e)
11 {
12 x=(x+y)/2;
13 y=n/x;
14 }
15 return x;
16 }
17 def main(args: Array[String]): Unit =
18 {
19 println("Enter a number: ")
20 var num:Int = scala.io.StdIn.readLine().toInt
21 println(squareRoot(num));
22 }
23 }
SquarerootBabylonian › squareRoot(n: Int)


Run SquarerootBabylonian
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
9
3
Process finished with exit code 0


Run SquarerootBabylonian
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
64
8
Process finished with exit code 0


Run SquarerootBabylonian
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
144
12
Process finished with exit code 0