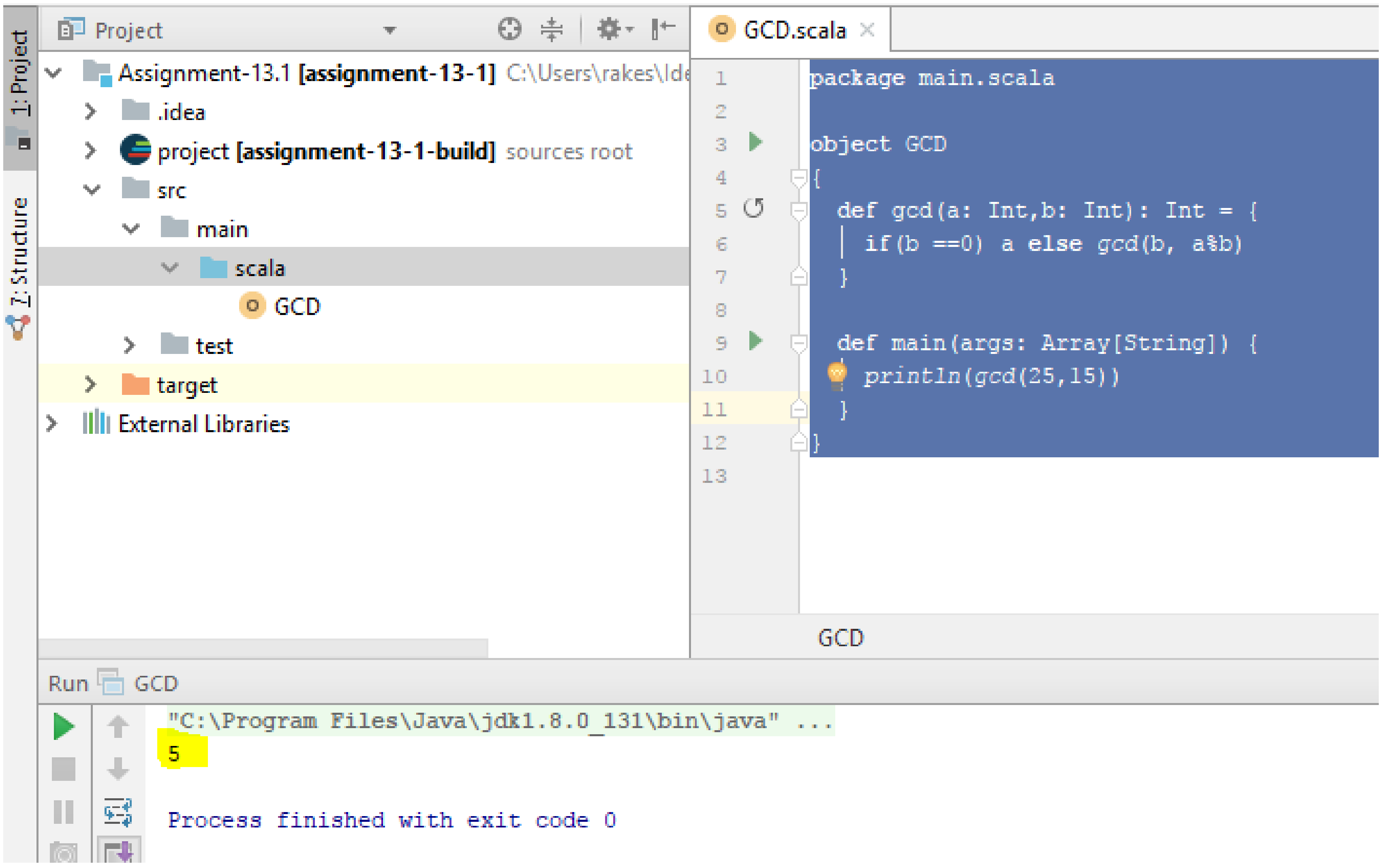
Problem Statement- **Create a Scala application to find the GCD of two numbers.**

Solution-

Below is the scala object written to find the GCD of two numbers-

```
package main.scala

object GCD
{
  def gcd(a: Int,b: Int): Int = {
    if(b ==0) a else gcd(b, a%b)
  }

  def main(args: Array[String]) {
    println(gcd(25,15))
  }
}
```

Below are the screenshots for the program written in INTELLIJ and the solution-



1: Project
7: Structure
Project
Assignment-13.1 [assignment-13-1] C:\Users\rakes\Ide
.idea
project [assignment-13-1-build] sources root
src
main
scala
GCD
test
target
External Libraries
GCD.scala
GCD
Run
GCD

```
package main.scala

object GCD
{
  def gcd(a: Int,b: Int): Int = {
    if(b ==0) a else gcd(b, a%b)
  }

  def main(args: Array[String]) {
    println(gcd(25,15))
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_131\bin\java" ...
5

Process finished with exit code 0
```

1. **Problem Statement-** **Write a Scala application to find the Nth digit in the sequence using standard for loop.**

Solution- Below is the code used for generating or extracting the nth digit of the Fibonacci sequence-

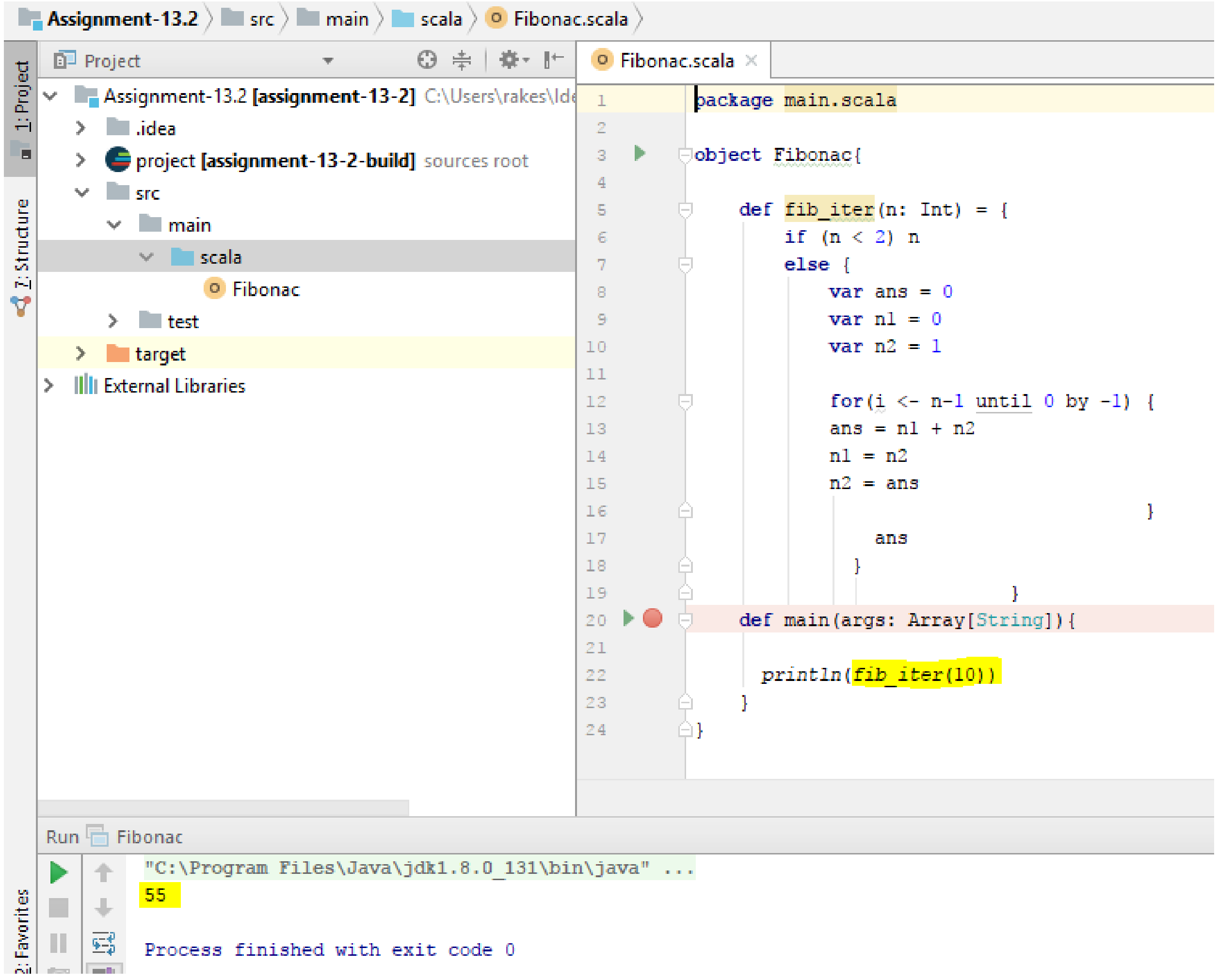
In below code we are defining a function named as fib_iter which will take a input of integer type. Then if the value of n is less than 2 it will return the same value else it willl iterate through for loop till n-1 summing the subsequent values.

For example we are taking 10 as n here. So it will sum below numbrs-

1,1,2,3,5,8,13,21,34 to give finally 55 as 10th number of this sequence-

```
package main.scala

object Fibonac{

    def fib_iter(n: Int) = {
        if (n < 2) n
        else {
            var ans = 0
            var n1 = 0
            var n2 = 1

            for(i <- n-1 until 0 by -1) {
            ans = n1 + n2
            n1 = n2
            n2 = ans
                                                      }

                ans
              }
                                        }
    def main(args: Array[String]){

      println(fib_iter(10))
    }
}
```

Below is the screenshot for the same with output-



```
package main.scala

object Fibonac{

    def fib_iter(n: Int) = {
        if (n < 2) n
        else {
            var ans = 0
            var n1 = 0
            var n2 = 1

            for(i <- n-1 until 0 by -1) {
            ans = n1 + n2
            n1 = n2
            n2 = ans
                                                      }
                ans
              }
                                        }
    def main(args: Array[String]){

      println(fib_iter(10))
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_131\bin\java" ...
55

Process finished with exit code 0
```

2. Problem Statement- Write a Scala application to find the Nth digit in the sequence using recursion.

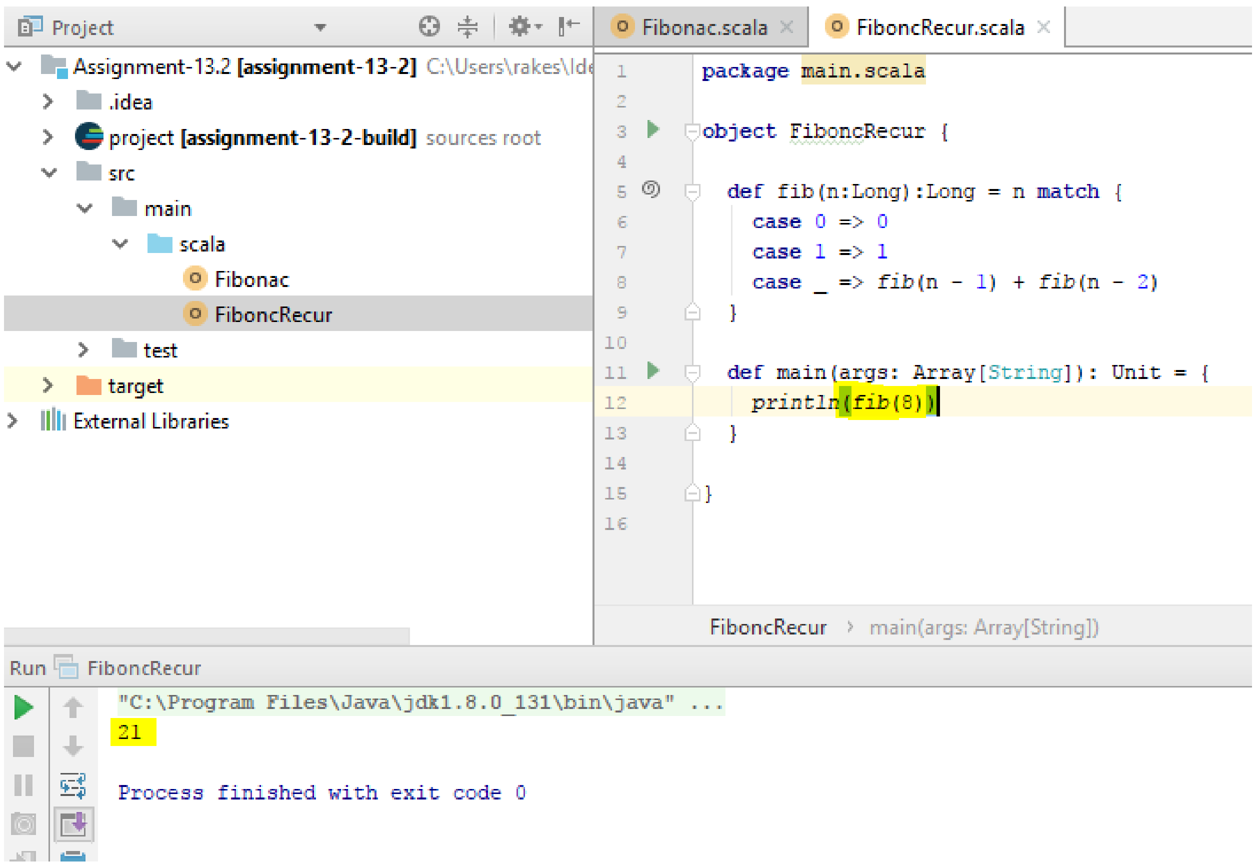
Solution- Below is the code used for generating or extracting the nth digit of the Fibonacci sequence using recursion-

In below code we are defining a function named as **fib** which will take a input of Long type. Then if the value of n is 0 or 1 then it will return 0 and 2 consecutively but if it is other than that then it will again call itself by passing values less than 1 and less than 2 than itself and summing them.
For example we are taking 8 as n here. So it will sum below numbrs-

1,1,2,3,5,8,13,21 to give finally 21 as 8th number of this sequence-

```
package main.scala

object FiboncRecur {

  def fib(n:Long):Long = n match {
    case 0 => 0
    case 1 => 1
    case _ => fib(n - 1) + fib(n - 2)
  }

  def main(args: Array[String]): Unit = {
    println(fib(8))
  }

}
```

Below screenshot shows the same with output-


Project
Assignment-13.2 [assignment-13-2] C:\Users\rakes\Ide
.idea
project [assignment-13-2-build] sources root
src
main
scala
Fibonac
FiboncRecur
test
target
External Libraries
Fibonac.scala
FiboncRecur.scala
FiboncRecur › main(args: Array[String])
Run
FiboncRecur

```
package main.scala

object FiboncRecur {

  def fib(n:Long):Long = n match {
    case 0 => 0
    case 1 => 1
    case _ => fib(n - 1) + fib(n - 2)
  }

  def main(args: Array[String]): Unit = {
    println(fib(8))
  }

}
```

```
"C:\Program Files\Java\jdk1.8.0_131\bin\java" ...
21

Process finished with exit code 0
```

Problem Statement- **Find square root of number using Babylonian method.**

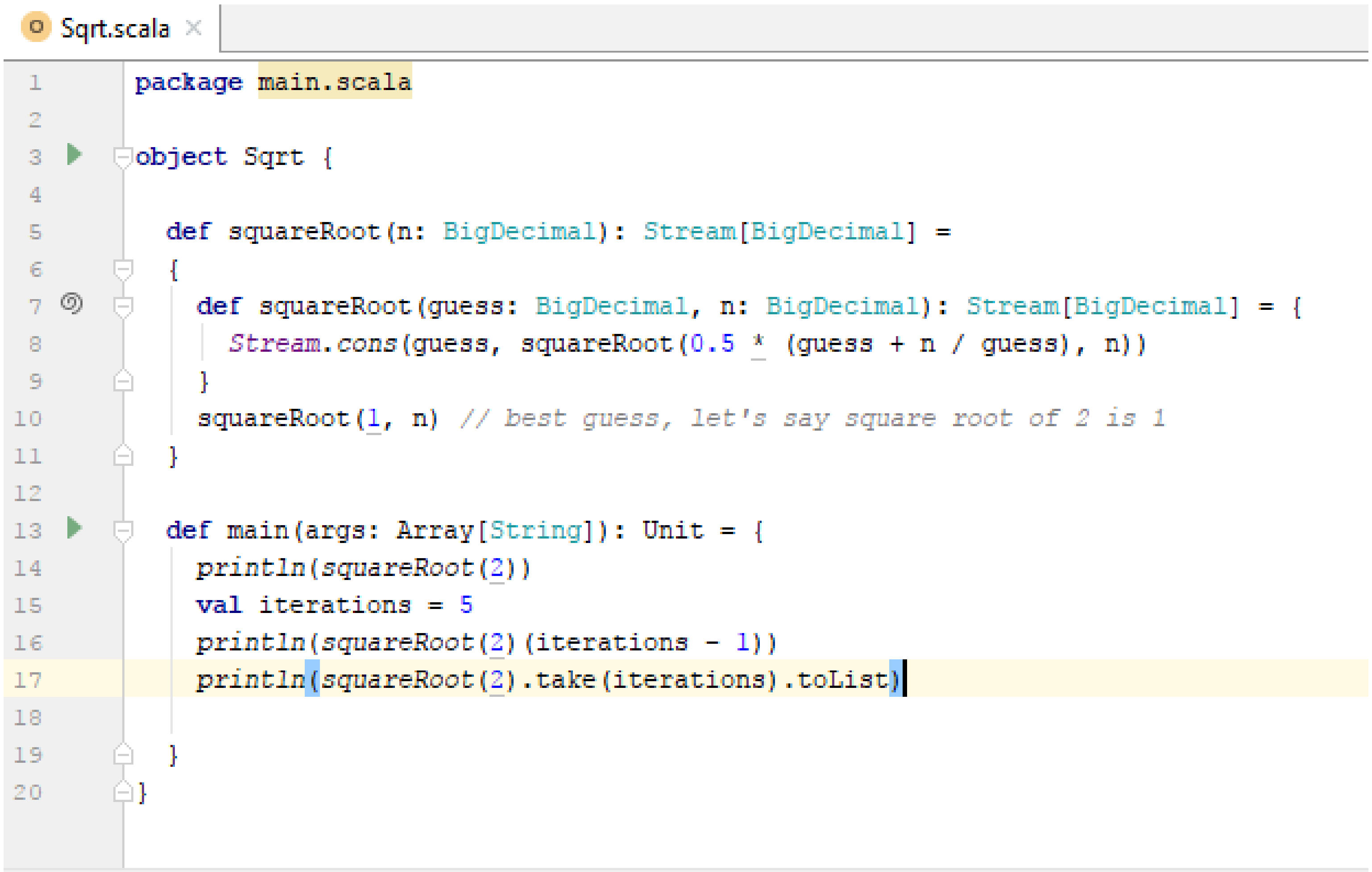
Solution-

Below is the code used to find the square root of number using Babylonian method –

Here we are trying to find the square root of 2 and chosen the value of iterations as 5.

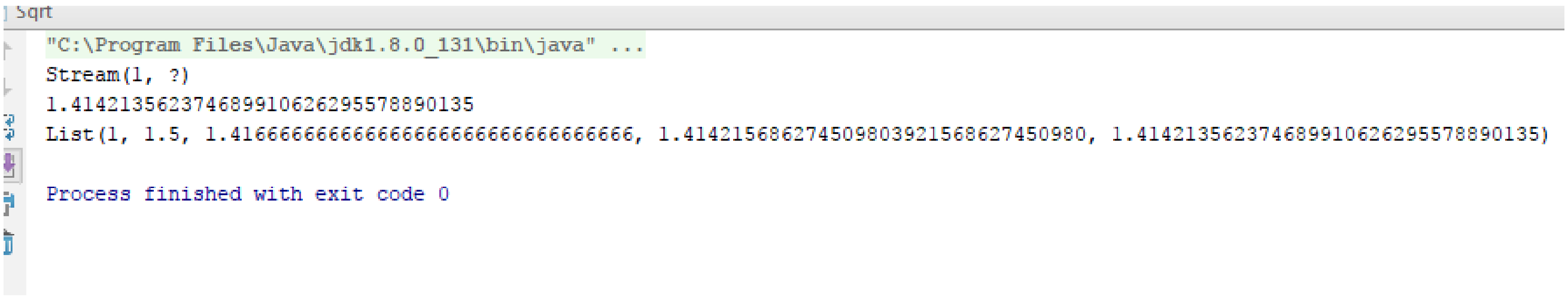
Now as the method suggests we are guessing 1 as the square root of number 2. So as approximation we are calculating using average as shown below-

```scala
package main.scala

object Sqrt {

  def squareRoot(n: BigDecimal): Stream[BigDecimal] =
  {
    def squareRoot(guess: BigDecimal, n: BigDecimal): Stream[BigDecimal] = {
      Stream.cons(guess, squareRoot(0.5 * (guess + n / guess), n))
    }
    squareRoot(1, n) // best guess, let's say square root of 2 is 1
  }

  def main(args: Array[String]): Unit = {
    println(squareRoot(2))
    val iterations = 5
    println(squareRoot(2)(iterations - 1))
    println(squareRoot(2).take(iterations).toList)

  }
}
```



```scala
package main.scala

object Sqrt {

  def squareRoot(n: BigDecimal): Stream[BigDecimal] =
  {
    def squareRoot(guess: BigDecimal, n: BigDecimal): Stream[BigDecimal] = {
      Stream.cons(guess, squareRoot(0.5 * (guess + n / guess), n))
    }
    squareRoot(1, n) // best guess, let's say square root of 2 is 1
  }

  def main(args: Array[String]): Unit = {
    println(squareRoot(2))
    val iterations = 5
    println(squareRoot(2)(iterations - 1))
    println(squareRoot(2).take(iterations).toList)

  }
}
```

Below screenshot shows the solution after running above code-



```
"C:\Program Files\Java\jdk1.8.0_131\bin\java" ...
Stream(1, ?)
1.414213562374689910626295578890135
List(1, 1.5, 1.416666666666666666666666666666666, 1.414215686274509803921568627450980, 1.414213562374689910626295578890135)

Process finished with exit code 0
```