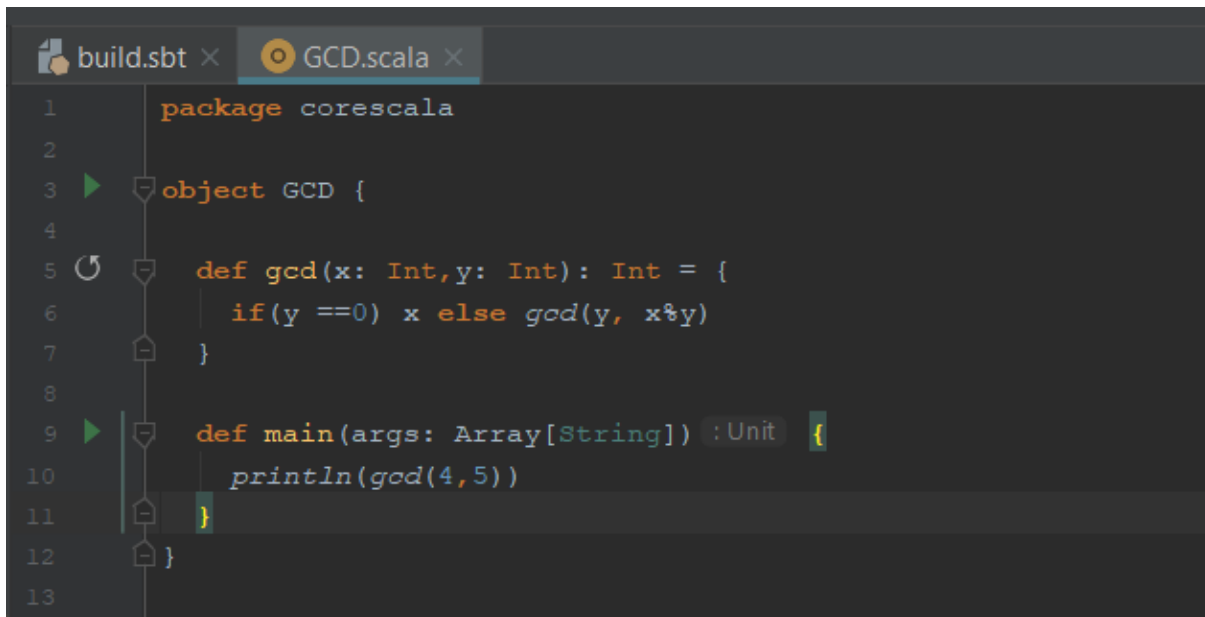


Assignment-15

Task 1:

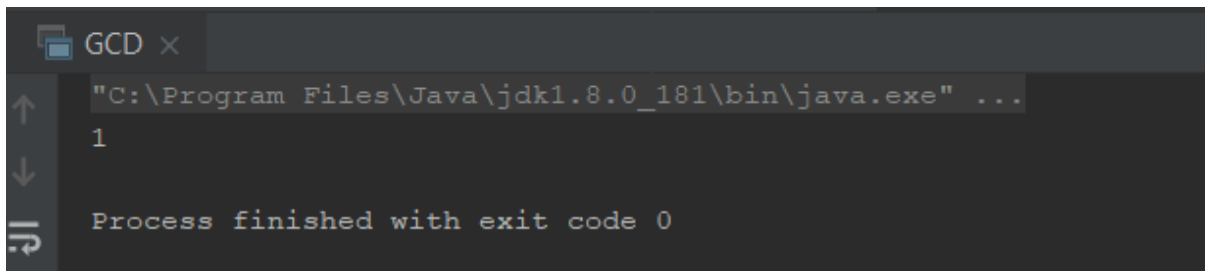
Create a Scala application to find the GCD of two numbers

The below screenshot shows the code for GCD of two numbers.



```
1 package corescala
2
3 object GCD {
4
5   def gcd(x: Int, y: Int): Int = {
6     if(y == 0) x else gcd(y, x%y)
7   }
8
9   def main(args: Array[String]) : Unit {
10     println(gcd(4,5))
11   }
12 }
13
```

The output of the above code is as shown in the below screenshot.



```
GCD x
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
1
Process finished with exit code 0
```

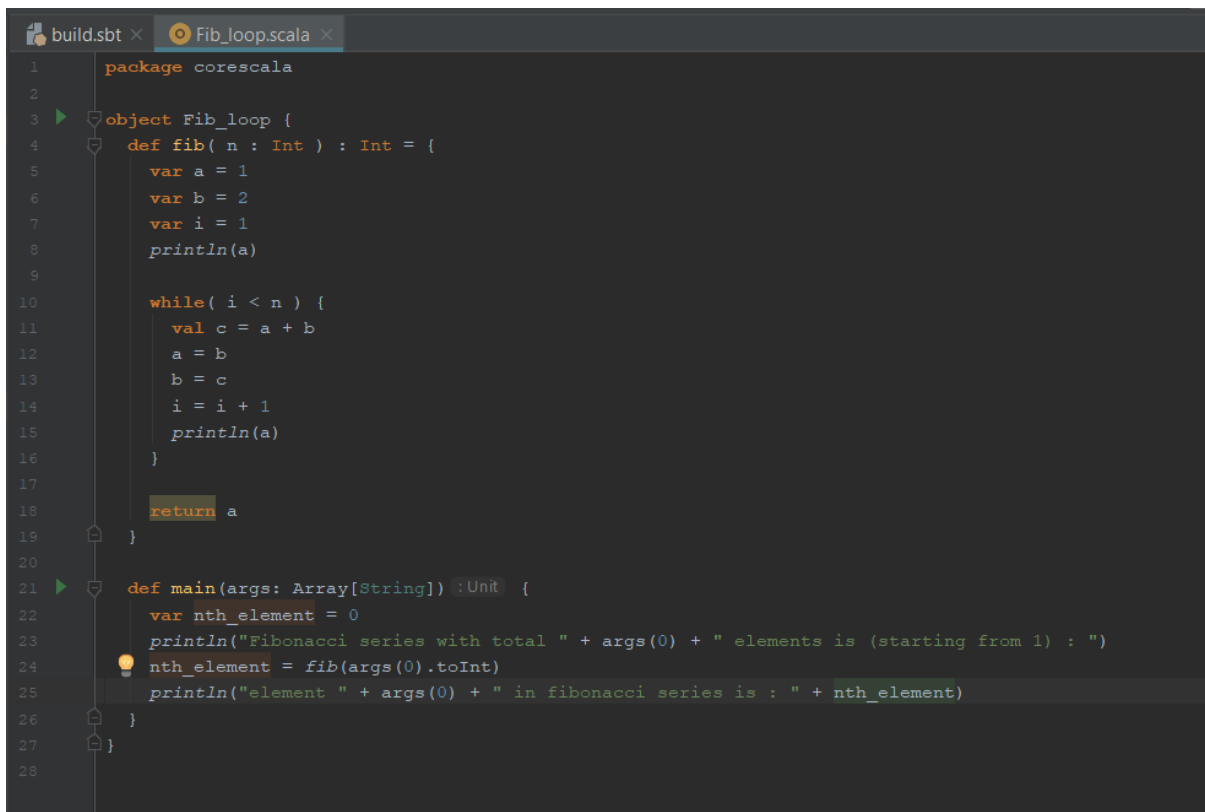
Task 2:

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.

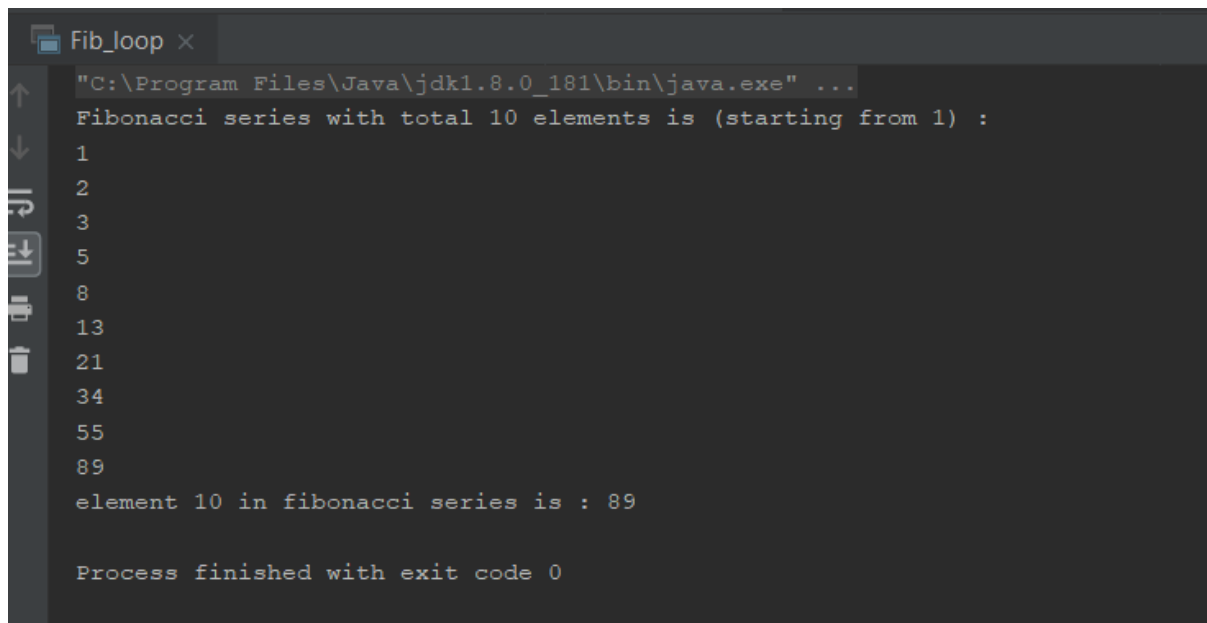
- Write the function using standard for loop

The scala code for the Fibonacci series using loop is shown in the below screenshot.



```
1  package corescala
2
3  object Fib_loop {
4    def fib( n : Int ) : Int = {
5      var a = 1
6      var b = 2
7      var i = 1
8      println(a)
9
10     while( i < n ) {
11       val c = a + b
12       a = b
13       b = c
14       i = i + 1
15       println(a)
16     }
17
18     return a
19   }
20
21   def main(args: Array[String]) :Unit {
22     var nth_element = 0
23     println("Fibonacci series with total " + args(0) + " elements is (starting from 1) : ")
24     nth_element = fib(args(0).toInt)
25     println("element " + args(0) + " in fibonacci series is : " + nth_element)
26   }
27 }
28
```

The output for the Fibonacci series using loop is shown in the below screenshot.

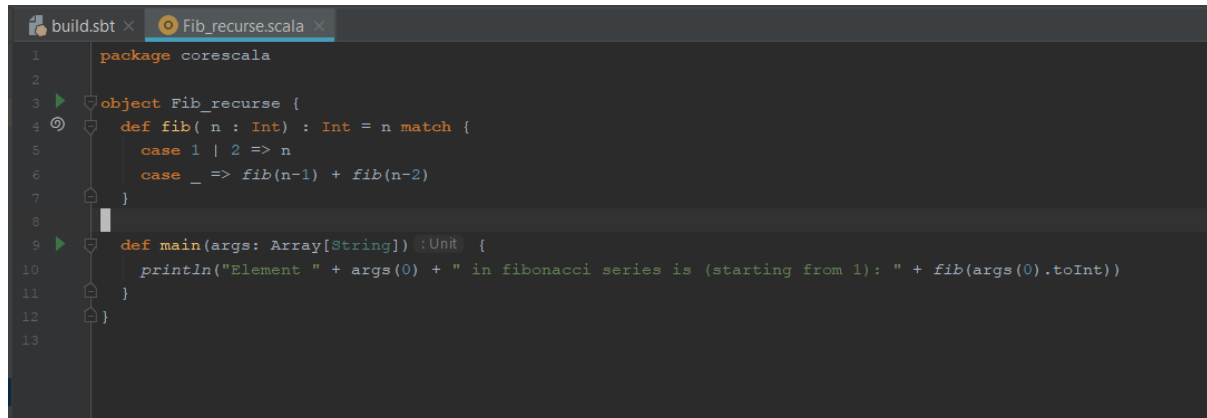


```
Fib_loop x
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Fibonacci series with total 10 elements is (starting from 1) :
1
2
3
5
8
13
21
34
55
89
element 10 in fibonacci series is : 89

Process finished with exit code 0
```

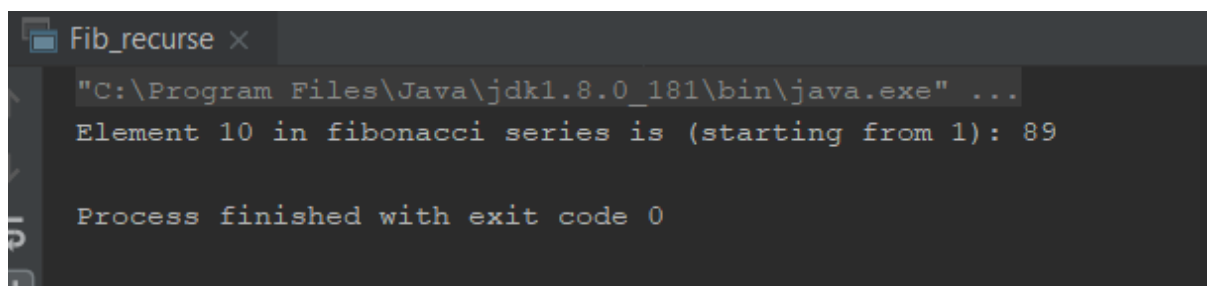
➤ Write the function using recursion

The scala code for Fibonacci series using recursion is as shown in the below screenshot.



```
build.sbt x  Fib_recurse.scala x
1 package corescala
2
3 object Fib_recurse {
4   def fib( n : Int) : Int = n match {
5     case 1 | 2 => n
6     case _ => fib(n-1) + fib(n-2)
7   }
8
9   def main(args: Array[String]) :Unit {
10    println("Element " + args(0) + " in fibonacci series is (starting from 1): " + fib(args(0).toInt))
11  }
12 }
13
```

The output for the Fibonacci series using recursion is as shown in the below screenshot.



```
Fib_recurse x
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Element 10 in fibonacci series is (starting from 1): 89

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).

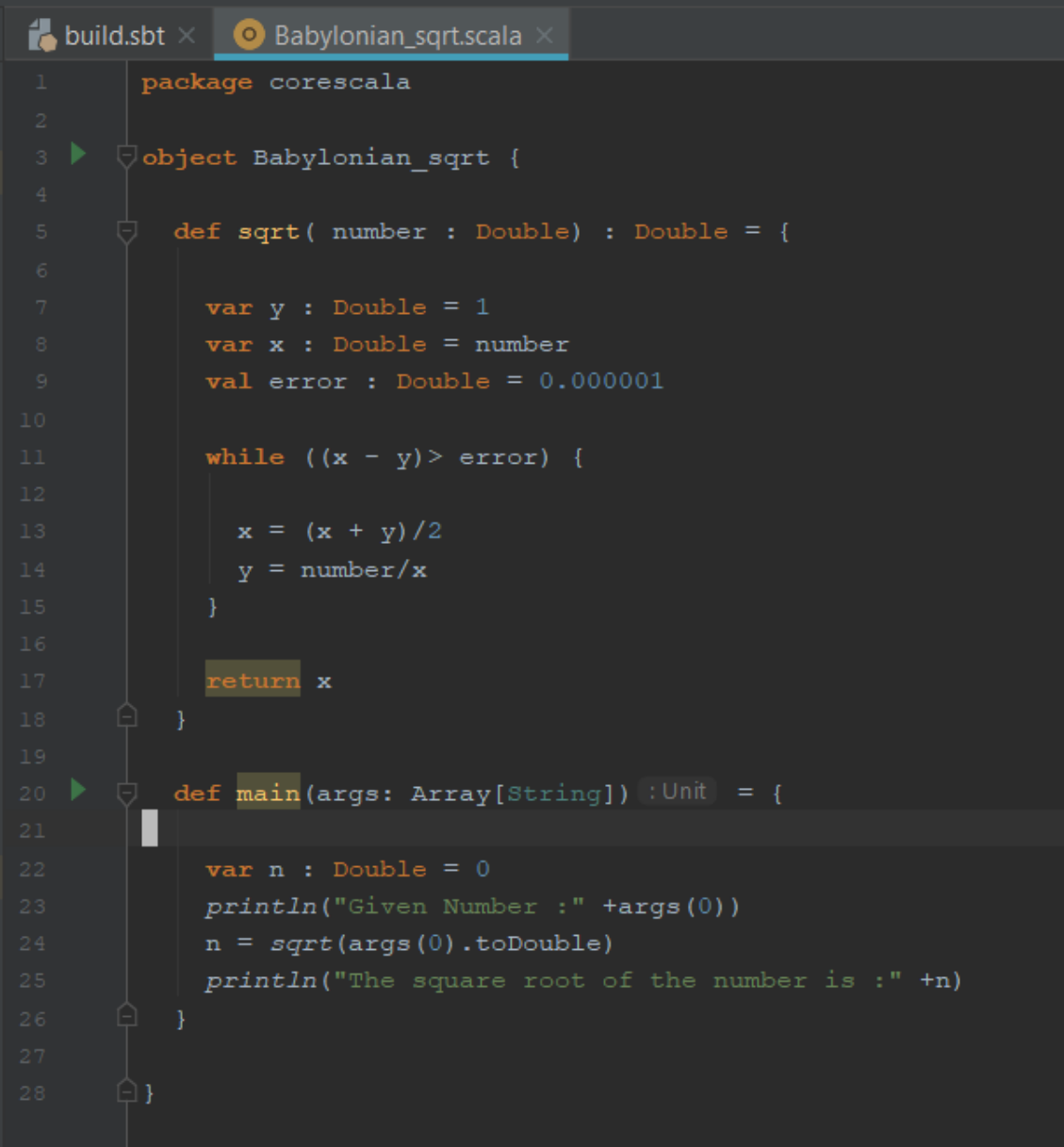
2. Initialize $y = 1$.

3. Do following until desired approximation is achieved.

a) Get the next approximation for root using average of x and y

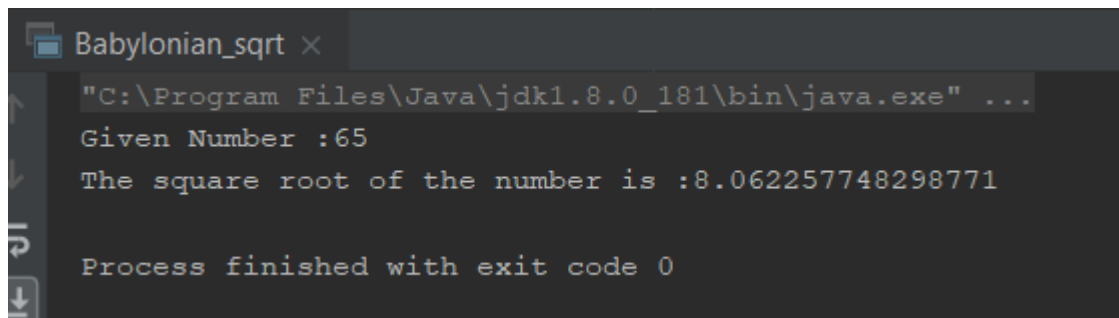
b) Set $y = n/x$

The code for Babylonian method to find square root of a number is as shown in the below screenshot.



```
1 package corescala
2
3 object Babylonian_sqrt {
4
5   def sqrt( number : Double) : Double = {
6
7     var y : Double = 1
8     var x : Double = number
9     val error : Double = 0.000001
10
11     while ((x - y) > error) {
12
13       x = (x + y) / 2
14       y = number / x
15     }
16
17     return x
18   }
19
20   def main(args: Array[String]) : Unit = {
21
22     var n : Double = 0
23     println("Given Number :" + args(0))
24     n = sqrt(args(0).toDouble)
25     println("The square root of the number is :" + n)
26   }
27
28 }
```

The output for the Babylonian method is as shown in the below figure.



```
Babylonian_sqrt x
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Given Number :65
The square root of the number is :8.062257748298771
Process finished with exit code 0
```