Session 15:

SCALA BASICS 2

Task 1

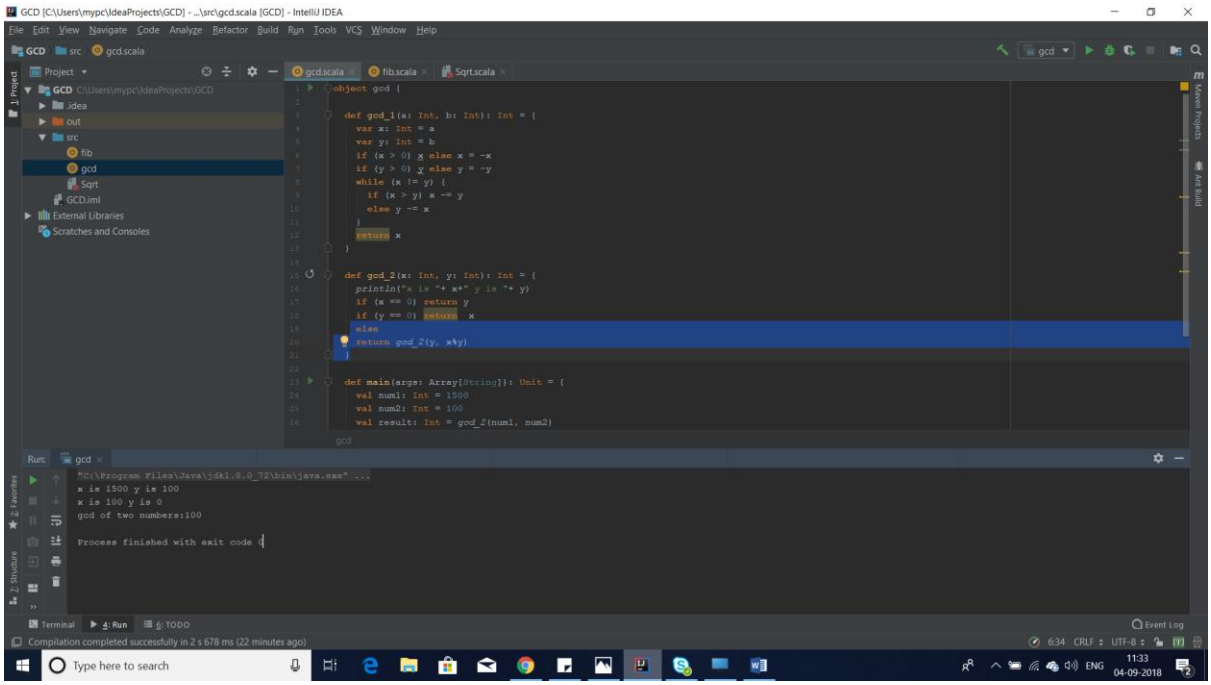
Create a Scala application to find the GCD of two numbers

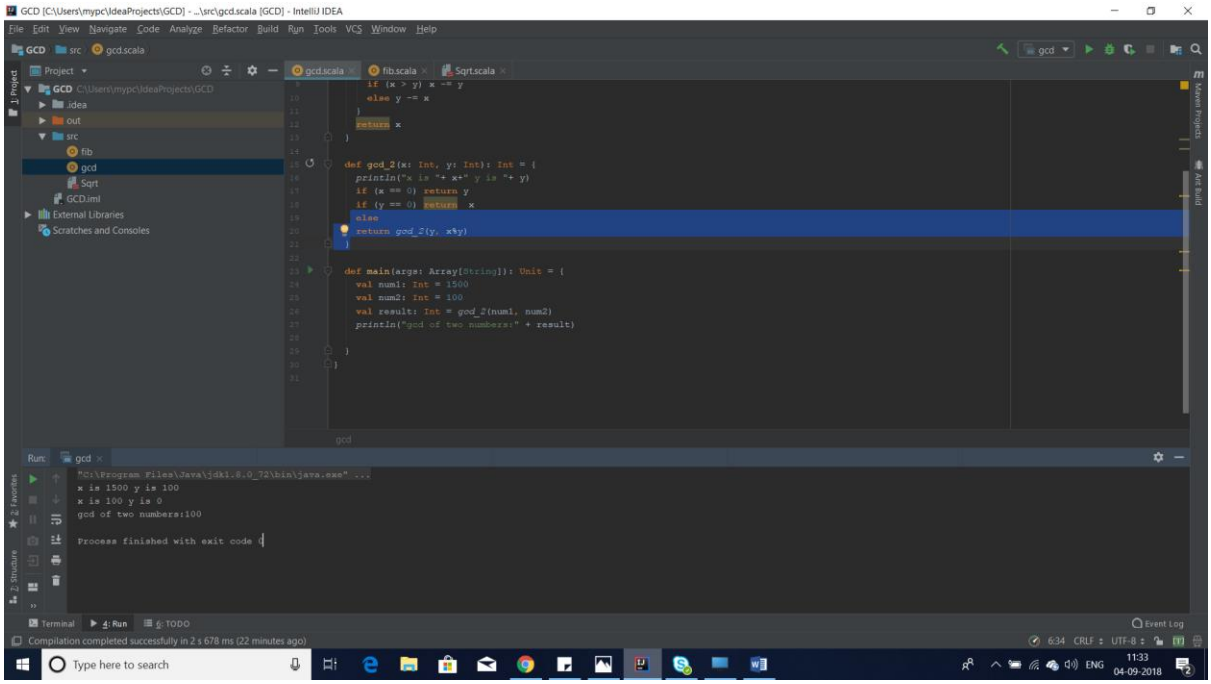
Solution: We can compute the gcd in the following ways:

```
Object gcd {
def gcd_1(a: Int,    b:Int): Int ={
var x : Int =a
var y :Int =b
if(x>0) x else x=-x
if(y>0)y else y=-y
while(x!=y){
if(x>y) x=x-y
else y=y-x
}
Return x
}

//Another way is:
def gcd_2(x:Int, b:int){
if(x==0) return y
if(y==0) return x
else return gcd_2(y, x%y)
}
Def main(args:Array[String]):unit{
val num1 :Int =1500
val num2 :Int= 100
val result :Int =gcd_2(num1,num2)
println("GCD  of two numbers is :"+ result)
```

}

}





Task 2

Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits.

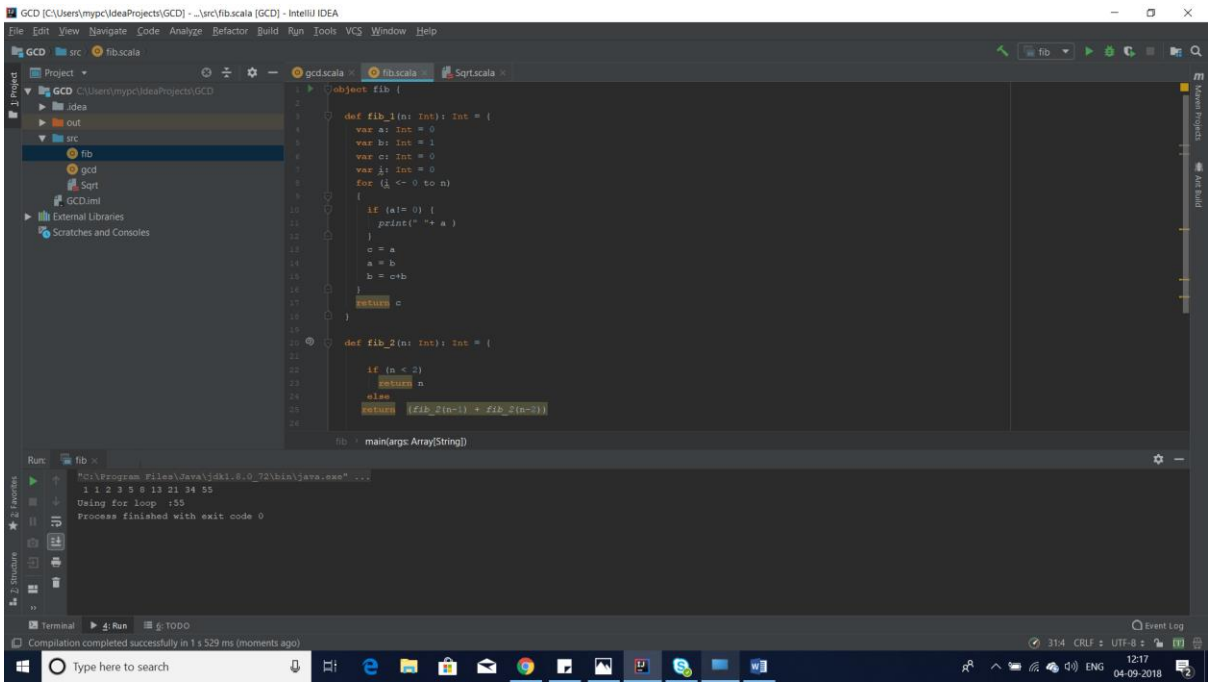
Write a Scala application to find the Nth digit in the sequence.

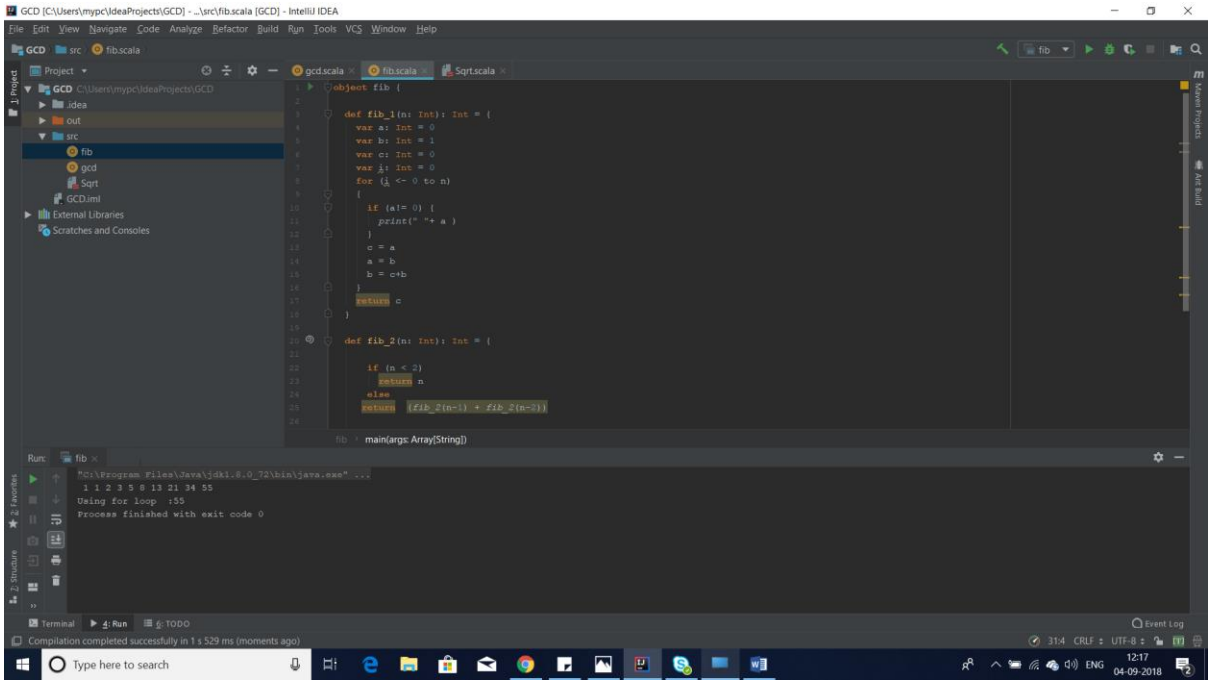
➢ Write the function using standard for loop

➢ Write the function using recursion

```
object fib {
def  fib_1(n:Int): Int ={
var a :Int = 0
var b :Int =1
var sum  :Int =0
var i :Int =0
for(i <- 0 to n){
if(a!=0){
print(“ “+a)
}
Sum =a+b
a=b
b=sum
}
Return sum
}
def main(args:Array[String]):Int ={
val num1 :Int =10
val result :Int = fib_1(num1)
println(“Using for loop:”+  fib_1(num1))
}
}
```

```
//Using Recursive  Function

def fib_2(n:Int): Int={

if(n<2) return n

else

  return  (fib_2(n-1) + fib_2(n-2))


}

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

Val num_1 : Int =10

Println(“ Fibonacci numbers using recursive :” +  fib_2(num_1))

}

}
```





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

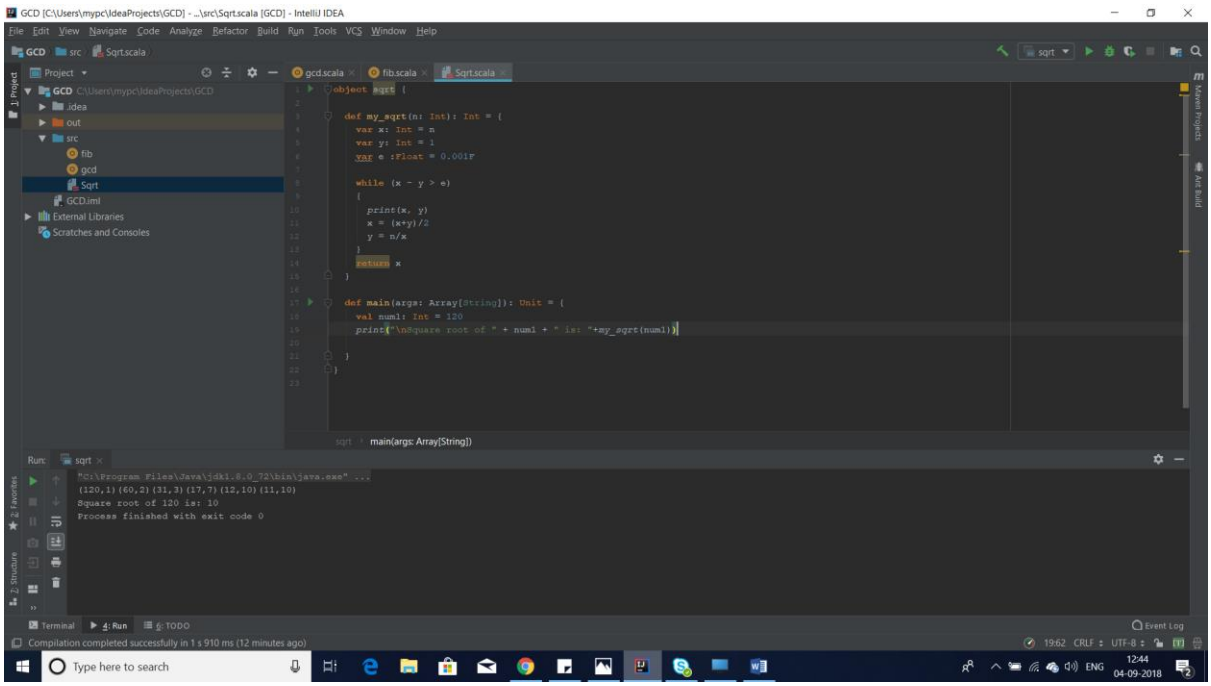
b) Set y = n/x

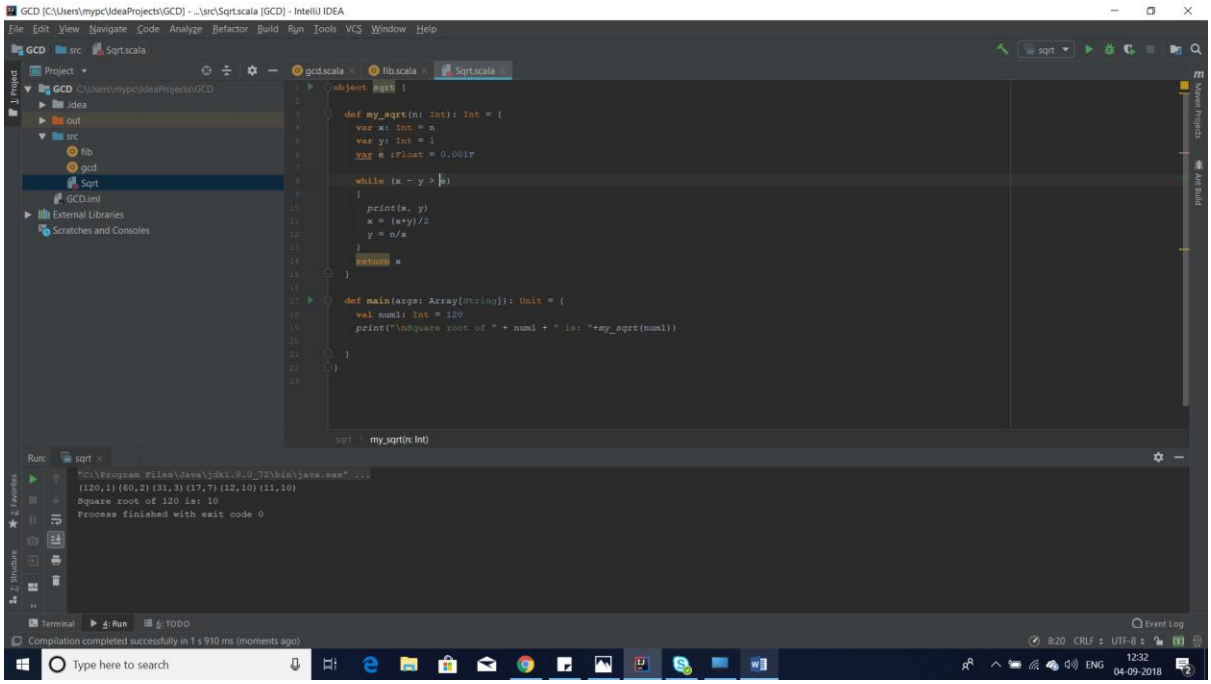
Solution :

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.

```
object sqrt {

def my_sqrt(n : Int) :Int ={


var x :Int =n
```

```
var y : Int =1

var e :Float =0.001F

while(x-y>e)


{

Println(x, y)

x= (x+y)/2

y=n/x

}

return x

}

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

val num_1 = 120

println("square root of num_1 is : "+  my_sqrt(num_1))

}

}
```




GCD [C:\Users\mypc\IdeaProjects\GCD] - ...\src\Sqrt.scala [GCD] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
GCD src Sqrt.scala
sqrt
Project
gcd.scala
fib.scala
Sqrt.scala
GCD C:\Users\mypc\IdeaProjects\GCD
.idea
out
src
fib
gcd
Sqrt
GCD.iml
External Libraries
Scratches and Consoles
object sqrt {
def my_sqrt(n: Int): Int = {
var x: Int = n
var y: Int = 1
var e :Float = 0.001f
while (x - y > e)
{
print(x, y)
x = (x+y)/2
y = n/x
}
return x
}
def main(args: Array[String]): Unit = {
val num1: Int = 120
print("\nSquare root of " + num1 + " is: "+my_sqrt(num1))
}
}
sqrt > my_sqrt(n: Int)
Run: sqrt
"C:\Program Files\Java\jdk1.8.0_72\bin\java.exe" ...
(120,1) (60,2) (31,3) (17,7) (12,10) (11,10)
Square root of 120 is: 10
Process finished with exit code 0
Terminal 4: Run TODO
Event Log
Compilation completed successfully in 1 s 910 ms (moments ago)
8:20 CRLF UTF-8
Type here to search
ENG
12:32
04-09-2018