BIG DATA HADOOP AND SPARK DEVLOPMENT

ASSIGNMENT 15

Table of Contents:

BIG DATA HADOOPAND SPARK DEVELOPMENT

1. Introduction

In this assignment, the given tasks are performed and Output of the tasks are recorded in the form of Screenshots.

2. Objective

This Assignment consolidates the deeper understanding of the Session – 15 SCALA BASICS 2

3. Problem Statement

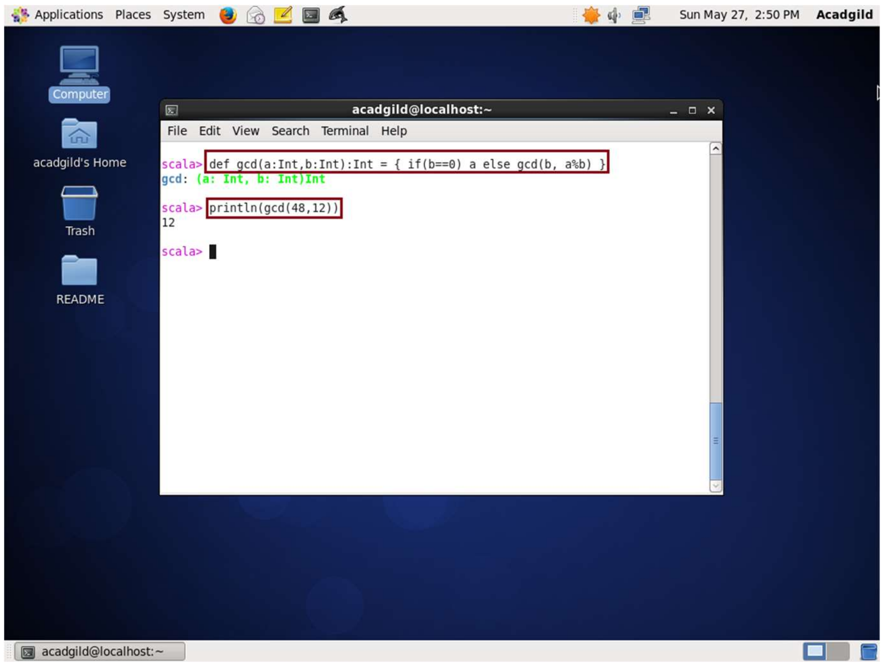
- Task 1
    - Create a Scala application to find the GCD of two numbers
- Task 2
    - Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits. Write a Scala application to find the Nth digit in the sequence.
    - Write the function using standard for loop
    - Write the function using recursion
- Task 3
    - Find square root of number using Babylonian method.
        - Start with an arbitrary positive start value x (the closer to the root, the better).
        - Initialize y = 1.
        - 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

4. Expected Output

- Task 1

Create a Scala application to find the GCD of two numbers

```
scala> def gcd(a:Int,b:Int):Int = { if(b==0) a else gcd(b,a%b) }

scala> println(gcd(48,12))
```


Applications Places System
Sun May 27, 2:50 PM Acadgild
Computer
acadgild's Home
Trash
README
acadgild@localhost:~
File Edit View Search Terminal Help
acadgild@localhost:~

```
scala> def gcd(a:Int,b:Int):Int = { if(b==0) a else gcd(b, a%b) }
gcd: (a: Int, b: Int)Int

scala> println(gcd(48,12))
12

scala> 
```

- Task 2

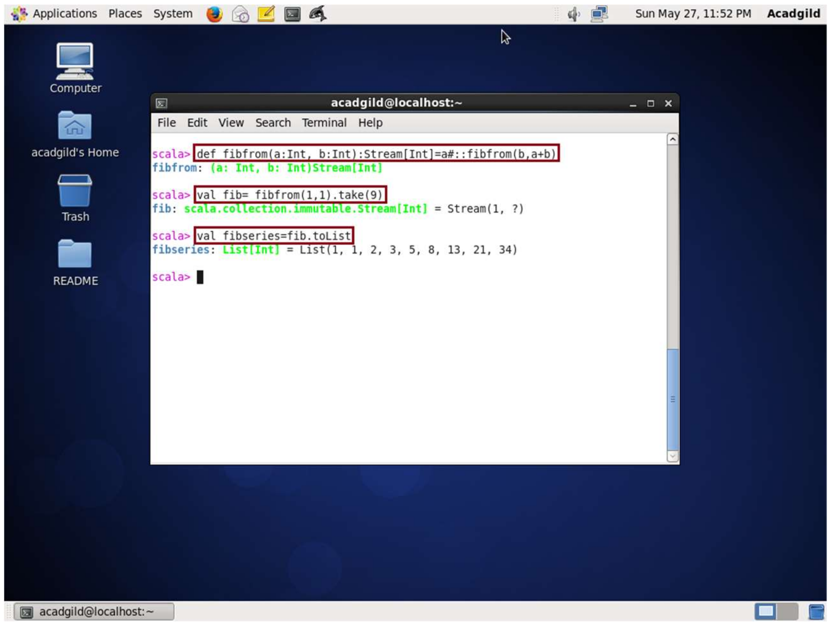
Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits. Write a Scala application to find the Nth digit in the sequence.

- Write the function using standard for loop

scala>def fibfrom(a:Int,b:Int): Stream[Int]= a #:: fibfrom(b, a+b)

scala> val fib =fibfrom(1,1). take(9)

scala> val fibseries=fib.toList


Applications Places System
Sun May 27, 11:52 PM Acadgild
Computer
acadgild's Home
Trash
README
acadgild@localhost:~
File Edit View Search Terminal Help
acadgild@localhost:~

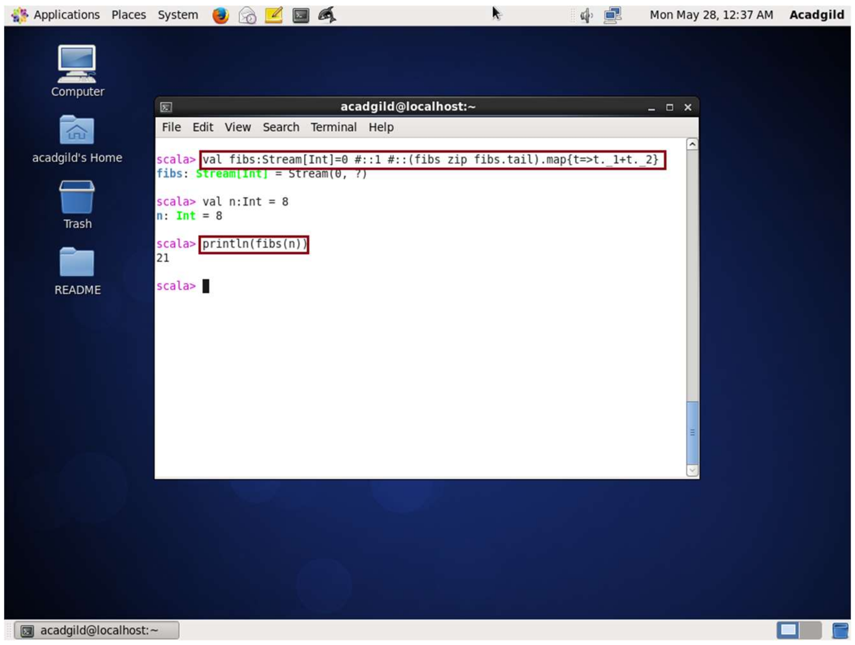
```
scala> def fibfrom(a:Int, b:Int):Stream[Int]=a#::fibfrom(b,a+b)
fibfrom: (a: Int, b: Int)Stream[Int]

scala> val fib= fibfrom(1,1).take(9)
fib: scala.collection.immutable.Stream[Int] = Stream(1, ?)

scala> val fibseries=fib.toList
fibseries: List[Int] = List(1, 1, 2, 3, 5, 8, 13, 21, 34)

scala> 
```

- Write the function using recursion

scala> val fibs: Stream[Int]=0 #:: 1 #:: (fibs zip fibs.tail).map{t=>t._1+t._2}

scala> val n: Int =8

scala> println(fibs(n))



```
scala> val fibs:Stream[Int]=0 #::1 #::(fibs zip fibs.tail).map{t=>t._1+t._2}
fibs: Stream[Int] = Stream(0, ?)

scala> val n:Int = 8
n: Int = 8

scala> println(fibs(n))
21

scala> 
```

- Task 3

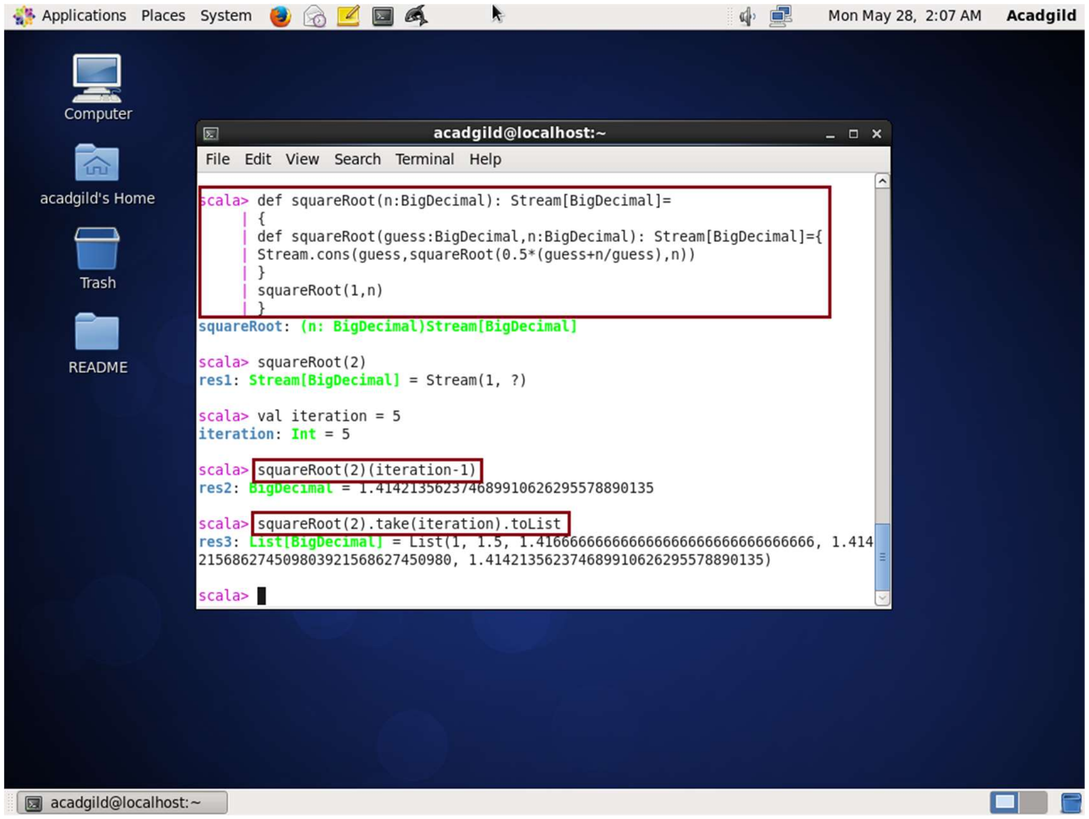
Find square root of number using Babylonian method.

- Start with an arbitrary positive start value x (the closer to the root, the better).
- Initialize y = 1.
- 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

```
scala> 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)

scala> squareRoot(2)

scala> val iterations = 5

scala> squareRoot(2)(iterations - 1)

scala> squareRoot(2).take(iterations).toList
```



Applications Places System
Mon May 28, 2:07 AM Acadgild
Computer
acadgild's Home
Trash
README
acadgild@localhost:~
File Edit View Search Terminal Help
acadgild@localhost:~

```
scala> 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)
     | }
squareRoot: (n: BigDecimal)Stream[BigDecimal]

scala> squareRoot(2)
res1: Stream[BigDecimal] = Stream(1, ?)

scala> val iteration = 5
iteration: Int = 5

scala> squareRoot(2)(iteration-1)
res2: BigDecimal = 1.414213562374689910626295578890135

scala> squareRoot(2).take(iteration).toList
res3: List[BigDecimal] = List(1, 1.5, 1.416666666666666666666666666666666, 1.414
215686274509803921568627450980, 1.414213562374689910626295578890135)

scala> 
```