ASSIGNMENT 15.1

Task 1
Create a Scala application to find the GCD of two numbers

SOLUTION

Basic Euclidean Algorithm for GCD
The algorithm is based on below facts.
If we subtract smaller number from larger (we reduce larger number), GCD doesn't change. So if we keep subtracting repeatedly the larger of two, we end up with GCD.
Now instead of subtraction, if we divide smaller number, the algorithm stops when we find remainder 0

```
  def Assignment15_1_task1=
  {

    // find GCD of two numbers
    println(GCD(25,15))
    def GCD(x:Int,y:Int):Int={

      if(y==0) x
      else
      GCD(y,x%y)
    }

  }
```

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

SOLUTION

```
def Assignment15_1_task2=
  {

    // fibonacci series using for loop
    def fib_1(n: Int) = {
      val s:Array[Int] = new Array(n+2)
      s(0)=0
      s(1)=1

      for(i<-2 to n)
        {

          s(i)=s{i-1}+s(i-2)
        }
      s(n)

    }
```

```
    // fibonacci series using the recursion
    def fib(n: Int): Int = {
      if (n <= 1) n
      else
        fib(n - 1) + fib(n - 2)
    }


  }
```

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

```
def Assignment15_1_task3=
{
  println("sqaure root of 50 using while loop  is= " +sqroot(50))
  println("sqaure root of 50 using recursion  is= " +sqrt(50))


  // square root using the while loop
  def sqroot(n:Float):Float=
  {
    var x:Float=n
    var y:Float=1
    val e:Float=0.000001f // error margin
    while((x-y)> e)
      {
        x=(x+y)/2
        y = n/x

      }
    x
  }
  // square root using the recursion
  def sqrt(n:Float):Float=
  {
    val e:Float=0.000001f // error margin
    def sqrt_1(x:Float):Float=
    {
      val y=n/x
      if((x-y) < e)
        x
      else
        sqrt_1((x+y)/2)
    }

    sqrt_1(n)

  }


}
```

Below is the complete source code and sample output of the program

```
package demo

object Assignments  {

 def main(args: Array[String]): Unit = {
   Assignment15_1_task1
   Assignment15_1_task2
   Assignment15_1_task3

 }

  def Assignment15_1_task1=
  {

    // find GCD of two numbers
    println("GCD for two numbers 25,15 is " +GCD(25,15))
    def GCD(x:Int,y:Int):Int={

      if(y==0) x
      else
      GCD(y,x%y)
    }

  }


  def Assignment15_1_task2=
  {

    println("9th Fibonacci number using for loop is= " +fib_1(9))
    println("9th Fibonacci number using recursion is= " +fib(9))

    // fibonacci series using for loop
    def fib_1(n: Int) = {
      val s:Array[Int] = new Array(n+2)
      s(0)=0
      s(1)=1

      for(i<-2 to n)
        {

          s(i)=s{i-1}+s(i-2)
        }
      s(n)

    }

    // fibonacci series using the recursion
    def fib(n: Int): Int = {
      if (n <= 1) n
      else
        fib(n - 1) + fib(n - 2)
    }


  }
```

```
def Assignment15_1_task3=
{
  println("sqaure root of 50 using while loop  is= " +sqroot(50))
  println("sqaure root of 50 using recursion  is= " +sqrt(50))


  // square root using the while loop
  def sqroot(n:Float):Float=
  {
    var x:Float=n
    var y:Float=1
    val e:Float=0.000001f // error margin
    while((x-y)> e)
      {
        x=(x+y)/2
        y = n/x

      }
    x


  }

  // square root using the recursion
  def sqrt(n:Float):Float=
  {
    val e:Float=0.000001f // error margin
    def sqrt_1(x:Float):Float=
    {
      val y=n/x
      if((x-y) < e)
        x
      else
        sqrt_1((x+y)/2)
    }

    sqrt_1(n)

  }


}
```

The output of the program is as below

GCD for two numbers 25,15 is 5
9th Fibonacci number using for loop is= 34
9th Fibonacci number using recursion is= 34
sqaure root of 50 using while loop is= 7.071068
sqaure root of 50 using recursion is= 7.071068

Process finished with exit code 0