



Task 1:

Problem Statement

Create a Scala application to find the GCD of two numbers.

Task: GCD of two numbers

Before going to create a SCALA application, we will just see the overview of GCD formula.

Greatest Common Divisor (GCD) of two or more integers, which are not all zero, is the largest positive integer that divides each of the integers.

For example, the **gcd of 8 and 12 is 4.**

Scala Application using IntelliJ

In the below scala code, we are going to find the gcd of the two numbers 12 and 8.

```
package Assignmane_13_1                                //package which we created

object GCD//a new object GCD is created
{
  defgcd(a: Int,b: Int): Int = { // declaring a function gcd
    if(b ==0) a else gcd(b, a%b)  2 integer variables a,b
  }
  defmain(args: Array[String]) //Our main function takes in a named
  {                               parameterargs which is an Array of String.
    println(gcd(12,8))           // print the result
  }
}
```



Required Output

```
Run GCD
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
4
Process finished with exit code 0
```

Screen Shot

```
Project
└─ Assignment_13_1 [assignment_13_1] D:\Abu\Technical
   └─ .idea
   └─ project [assignment_13_1-build] sources root
   └─ src
      └─ main
         └─ scala
            └─ Assignmane_13_1
               └─ GCD
                  └─ GCD.scala

Run GCD
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
4
Process finished with exit code 0
```

```
1 package Assignmane_13_1
2
3 object GCD
4 {
5   def gcd(a: Int, b: Int): Int = {
6     if (b == 0) a else gcd(b, a % b)
7   }
8   def main(args: Array[String]) {
9     println(gcd(12, 8))
10  }
11 }
12
```

Task 2:

A 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

Before going in to the tasks, we will just see an over view that what is the Fibonacci number,

The Fibonacci sequence is the series of numbers,

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

The next number is found by adding up the two numbers before it.

The 2 is found by adding the two numbers before it (1+1)

The 3 is found by adding the two numbers before it (1+2),

And the 5 is (2+3),

And so on!

Example: the next number in the sequence above is $21+34 = 55$

n =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
xn =	0	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987

Formula,

$$x_n = x_{n-1} + x_{n-2}$$

Example,

The 8th term is the 7th term plus the 6th term: $X_8 = X_7 + X_6$

From the above table,

The 8th term is 21, hence the 7th term 21+the 6th term 13 = 34.

Task 1: write function using standard for loop

Scala code

```
package Assignment13_2

object fibseries
{
  def main(args: Array[String]): Unit = {
    println("Enter a number: ")
    var num: Int = scala.io.StdIn.readLine().toInt

    var n1 = 0
    var n2 = 1

    var a: Int = 0;
    var b: Int = 0;
```



Screen Shot:

```
1 package Assignment13_2
2
3 object fibseries
4 {
5   def main(args: Array[String]): Unit = {
6
7     println("Enter a number: ")
8     var num: Int = scala.io.StdIn.readLine().toInt
9
10    var n1=0
11    var n2=1
12
13    var a: Int=0;
14    var b: Int=0;
15
16    println("Standard For loop")
17    for(a <- 1 to num){
18      val sumOfPrevTwo = n1+n2
19      n1=n2
20      n2 = sumOfPrevTwo
21    }
22    println(num + "nth digit in the sequence is:" + n2)
23  }
24 }
25
```

Run fibseries

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
8
Standard For loop
8nth digit in the sequence is:34
Process finished with exit code 0
```

Output

When we provide number 8 as input, the 8th digit in the Fibonacci sequence is 34.



```
Run fibseries
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
8
Standard For loop
8nth digit in the sequence is:34
Process finished with exit code 0
```

If we give the input as 10, the 10th digit of Fibonacci sequence is 89

```
Run fibseries
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
10
Standard For loop
10nth digit in the sequence is:89
Process finished with exit code 0
|
```

Task2 - Write the function using recursion

Scala code

```
object fibseriesrecursion
{
  def main(args: Array[String]): Unit = {
    println("Enter a number: ")
    var num: Int = scala.io.StdIn.readLine().toInt
    println("Using Recursion")
    println(num + "nth digit in the sequence is: " + fib(num))

    def fib(n: Int): Int =
      if (n < 2)
        1
      else
        fib(n-1) + fib(n-2)
  }
}
```

Screen shot:



The screenshot shows the IntelliJ IDEA IDE. The top toolbar includes icons for Project, Run, Debug, Settings, and Help. The Project view on the left shows the hierarchy: Assignment_13_2 [assignment_13_2] > .idea > project [assignment_13_2-build] sources root > src > main > scala > Assignment13_2 > fibseries > fibseriesrecursion. The main editor displays the code for fibseriesrecursion.scala:

```
1 object fibseriesrecursion
2 {
3   def main(args: Array[String]): Unit = {
4
5     println("Enter a number: ")
6     var num: Int = scala.io.StdIn.readLine().toInt
7     println("Using Recursion")
8     println(num + "nth digit in the sequence is: " + fib(num))
9
10    def fib(n: Int): Int =
11      if (n < 2)
12        1
13      else
14        fib(n-1+fib(n-2))
15    }
16  }
17 }
```

The Run view at the bottom shows the execution of the fibseries program. The command used is "C:\Program Files\Java\jdk1.8.0_144\bin\java" ... The output is:

```
Enter a number:
10
Standard For loop
10th digit in the sequence is:89

Process finished with exit code 0
```

Output

The screenshot shows the Run view of the IntelliJ IDEA IDE. The command used is "C:\Program Files\Java\jdk1.8.0_144\bin\java" ... The output is:

```
Enter a number:
10
Standard For loop
10th digit in the sequence is: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 theRoot, 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 Babylonian method for finding square roots involves dividing and averaging, over and over, to obtain a more accurate solution with each repeat of the process. Step 2: Divide your original number by your guess. Step 3: Find the average of these numbers. Step 4: Use this average as your next guess.

Task – Find square root of a number using Babylonian Method

Scala code

```
package Assignment13_3

object SquarerootBabylonian
{
  def squareRoot(n: Int): Int =
  {
    var x = n;
    var y = 1;
    var e = 0.000001;
    while (x - y > e)
    {
      x = (x + y) / 2;
      y = n / x;
    }
    return x;
  }
  def main(args: Array[String]): Unit =
  {
    println("Enter a number: ")
    var num: Int = scala.io.StdIn.readLine().toInt
    println(squareRoot(num));
  }
}
```



Screen Shot

The screenshot shows an IDE with a project named 'Assignment13_3'. The file 'SquarerootBabylonian.scala' is open, showing the following code:

```
1 package Assignment13_3
2
3 object SquarerootBabylonian
4 {
5     def squareRoot(n: Int): Int =
6     {
7         var x = n;
8         var y = 1;
9         var e = 0.000001;
10        while (x - y > e)
11        {
12            x = (x + y) / 2;
13            y = n / x;
14        }
15        return x;
16    }
17
18    def main(args: Array[String]): Unit =
19    {
20        println("Enter a number: ")
21        var num: Int = scala.io.StdIn.readLine().toInt
22        println(squareRoot(num));
23    }
24 }
```

The 'Run' console at the bottom shows the execution of the program:

```
Run SquarerootBabylonian
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
9
3
Process finished with exit code 0
```

Output

If we enter a number 64, the square root of that value is 8

The screenshot shows the 'Run' console with the following output:

```
Run SquarerootBabylonian
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
64
8
Process finished with exit code 0
```

Find square root of number using Babylonian method.

1. Start with an arbitrary positive start value x (the closer to theRoot, 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 Babylonian method for finding square roots involves dividing and averaging, over and over, to obtain a more accurate solution with each repeat of the process. Step 2: Divide your original number by your guess. Step 3: Find the average of these numbers. Step 4: Use this average as your next guess.

Task – Find square root of a number using Babylonian Method

Scala code

```
package Assignment13_3

object SquarerootBabylonian
{
  def squareRoot(n: Int): Int =
  {
    var x = n;
    var y = 1;
    var e = 0.000001;
    while (x - y > e)
    {
      x = (x + y) / 2;
      y = n / x;
    }
    return x;
  }
  def main(args: Array[String]): Unit =
  {
    println("Enter a number: ")
    var num: Int = scala.io.StdIn.readLine().toInt
    println(squareRoot(num));
  }
}
```



Screen Shot

The screenshot shows an IDE with a project named 'Assignment13_3'. The file explorer on the left shows the project structure, including 'src/main/scala/Assignment13_3/SquarerootBabylonian.scala'. The main editor displays the Scala code for the 'SquarerootBabylonian' object. The code defines a 'squareRoot' function that uses the Babylonian method to calculate the square root of a number 'n'. The function iteratively refines the value of 'x' until the difference between 'x' and 'y' is less than a small epsilon 'e'. The 'main' function prompts the user to enter a number and prints the result of the 'squareRoot' function.

```
1 package Assignment13_3
2
3 object SquarerootBabylonian
4 {
5     def squareRoot(n: Int): Int =
6     {
7         var x = n;
8         var y = 1;
9         var e = 0.000001;
10        while (x - y > e)
11        {
12            x = (x + y) / 2;
13            y = n / x;
14        }
15        return x;
16    }
17
18    def main(args: Array[String]): Unit =
19    {
20        println("Enter a number: ")
21        var num: Int = scala.io.StdIn.readLine().toInt
22        println(squareRoot(num));
23    }
24 }
```

The Run console at the bottom shows the execution of the program. It displays the command to run the program, the prompt 'Enter a number:', the input '64', the output '8', and the message 'Process finished with exit code 0'.

Output

If we enter a number 64, the square root of that value is 8

The screenshot shows the Run console for the 'SquarerootBabylonian' program. It displays the command to run the program, the prompt 'Enter a number:', the input '64', the output '8', and the message 'Process finished with exit code 0'.