

Assignment 15 Scala 2

Task 1

Create a Scala application to find the GCD of two numbers

Source Code:

```
class appGCD {

  /**Method to find the GCD of 2 numbers***/
  def gcd(a: Int, b: Int): Int = {
    if(b == 0) a else gcd(b, a%b)
  }

  /**Method to display list of choices to the user***/
  def OptionsList(): Unit = {
    println("\nGCD of 2 numbers")
    println("-----")

    println("\nSelect one of the following:")
    println("1. Compute GCD with command line argument")
    println("2. Compute GCD with standard input argument")

    println("\nEnter your choice (1 or 2): ")

  }
}

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

    var wish = ""

    /**Creating the instance of the appGCD class***/

    val aGCD = new appGCD()

    do {

      /**Calling the method to display the list of options to the user***/
      aGCD.OptionsList()

      val choice = scala.io.StdIn.readLine()

      /**Find GCD from CommandLine Input Arguments (Get from the user)***/

      if (choice.toInt == 1) {
```

```

val input1 = args(0).toInt
val input2 = args(1).toInt

println("\nCMD: GCD of ${input1} and ${input2} is : " + aGCD.gcd(input1, input2))

}

/**Find GCD from Standard Input Arguments (Get from the user)*/
else if (choice.toInt == 2) {

println("Enter the 1st number : ")

val inp1 = scala.io.StdIn.readLine().toInt

println("Enter the 2nd number : ")
val inp2 = scala.io.StdIn.readLine().toInt

println("STDIN: GCD of ${inp1} and ${inp2} is : " + aGCD.gcd(inp1, inp2))
}
else {

println("Invalid choice!")

}

/**DoWhile loop conditional variable*/
println("\nDo you wish to continue? (Y/N) : ")
wish = scala.io.StdIn.readLine().toUpperCase

println("-----\n")
}
while (wish.equals("Y"))
}

}

```

Output:

```
GCD of 2 numbers
-----

Select one of the following:
1. Compute GCD with command line argument
2. Compute GCD with standard input argument

Enter your choice (1 or 2):
2
Enter the 1st number :
25
Enter the 2nd number :
2
STDIN: GCD of ${inp1} and ${inp2} is : 1

Do you wish to continue? (Y/N) :
N
-----
```

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*

Source Code:

```
import scala.annotation.tailrec

object appFiboSeries {

  def recFibonacci(n: Int, nth: Int): Unit = {
    var concat_result = "1"

    /**Method to find out the Fibonacci Series using Recursion***/

    @tailrec def fiboRecursive(n: Int, prev: BigInt = 0, next: BigInt = 1): BigInt = n match {

      case 0 => prev
```

```
case 1 => next
```

```
case _ =>
```

```
concat_result = concat_result + (prev + next)
```

```
fibRecursive(n - 1, next, next + prev)
```

```
}
```

```
fibRecursive(n)
```

```
get_nthchar_and_print(n, concat_result, nth)
```

```
}
```

```
/**Method to find out the Fibonacci Series using For Loop***/
```

```
def LoopFibo(n: Int, nth: Int): Unit = {
```

```
var concat_result = "1"
```

```
if (n < 2) {
```

```
println(n)
```

```
}
```

```
else {
```

```
var result: BigInt = 0
```

```
var n1: BigInt = 0
```

```
var n2: BigInt = 1
```

```
for (i <- 1 until n) {
```

```
result = n1 + n2
```

```
n1 = n2
```

```
n2 = result
```

```
concat_result = concat_result + result
```

```
}
```

```
get_nthchar_and_print(n, concat_result, nth)
```

```
result
```

```
}
```

```
}
```

```
/**Method to display Nth character in the Fibonacci Sequence***/ def
```

```
get_nthchar_and_print(n: Int, seq: String, nth: Int): Unit = {
```

```

println(s"The Fibonacci Series ($n): " + seq)

println(s"The digit at the place $nth of Fibo Sequence ($n): " + seq.charAt(nth - 1).toChar)

}

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

var wish = ""

println("Fibonacci Series")

println("-----")

do {
println("Select one of the following:")
println("1. Find Nth digit in the Fibonacci Series using For Loop")

println("2. Find Nth digit in the Fibonacci Series using Recursion")

println("Enter your choice (1 or 2): ")

var choice = scala.io.StdIn.readLine()

println("Enter the number of digits for Fibonacci Sequence: ")

var digits: Int = scala.io.StdIn.readLine().toInt

println("Enter the Nth digit to be found in the Fibonacci Sequence: ")

var nthFind: Int = scala.io.StdIn.readLine().toInt

println("-----")

if (choice.toInt == 1) {

/**Call to method "LoopFibo" to find out the Fibonacci Series using For Loop***/

println(s"Fibonacci Series using For Loop:")

println("-----")

LoopFibo(digits, nthFind)

}

/**Call to method "recFibonacci" to find out the Fibonacci Series using Recursion***/

```

```
else if (choice.toInt == 2) { println(s"Fibonacci Series using Recursion:")
println("-----")

recFibonacci(digits, nthFind)

}
else {

println(s"Invalid Choice!")

}

println("Do you wish to continue? (Y/N): ")

/**Do-While Loop for condition variable**/ wish = scala.io.StdIn.readLine().toUpperCase

println("-----\n")
}

while (wish.equals("Y"))

}

}
```

Output

```
appFiboSeries$ [Scala Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (05-Dec-2018, 11:46:02 PM)
Fibonacci Series
-----
Select one of the following:
1. Find Nth digit in the Fibonacci Series using For Loop
2. Find Nth digit in the Fibonacci Series using Recursion
Enter your choice (1 or 2):
1
Enter the number of digits for Fibonacci Sequence:
18
Enter the Nth digit to be found in the Fibonacci Sequence:
2
-----
Fibonacci Series using For Loop:
-----
The Fibonacci Series (18): 112358132134558914423337761098715972584
The digit at the place 2 of Fibo Sequence (18): 1
Do you wish to continue? (Y/N):
Y
-----
Select one of the following:
1. Find Nth digit in the Fibonacci Series using For Loop
2. Find Nth digit in the Fibonacci Series using Recursion
Enter your choice (1 or 2):
2
Enter the number of digits for Fibonacci Sequence:
13
Enter the Nth digit to be found in the Fibonacci Sequence:
3
-----
Fibonacci Series using Recursion:
-----
The Fibonacci Series (13): 1123581321345589144233
The digit at the place 3 of Fibo Sequence (13): 2
Do you wish to continue? (Y/N):
```

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 $y = n/x$*

Source Code

```

/**Dividing and Averaging Method to calculate square root of a number***/ object
appSqRootBM {

/**Function to return square root of a number using Babylonian Method***/ def
squareRootBM(num: Int): Float = {

/**Arbitrary positive value x from the user***/ var x: Float = num

/**Initialize y***/

var y: Float = 1

/**e decides the accuracy level***/

/**This is checked when we aren't sure if the number is a perfect square***/ val e: Double =
0.000001

/**Performs division and averaging until the accuracy level***/ while(x - y > e) {

x = (x + y) / 2
y = num / x
}
x /**Returns the square root value***/

}

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

var wish = ""

println("\nSquare Root using Babylonian Method")

println("-----")

do {

println("\nEnter the number: ")

var input = scala.io.StdIn.readLine().toInt

/**Calls the function to calculate Square Root using Babylonian Method***/
println("\n_____")
println(s"Square Root of $input is ${squareRootBM(input)}")

println("\n_____")

println("\n\nDo you wish to continue? (Y/N) : ")

/**Do-While Loop for condition variable***/ wish = scala.io.StdIn.readLine().toUpperCase

```



```
println("-----\n")
}
while (wish.equals("Y"))
{
}
}
```

OutPut

```
CONSOLE
<terminated> appSqRootBM$ [Scala Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (05-Dec-2018, 11:50:50 PM)

Square Root using Babylonian Method
-----

Enter the number:
5

-----
Square Root of 5 is 2.236068

-----

Do you wish to continue? (Y/N) :
N
|-----
```