# Scala -2 - assignmement(Scala)

Import Notebook

```
/**Task 1
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.
*/
//o Write the function using standard for loop
def fib1(n:Int):Int={

  var first = 0
  var second = 1
  var count = 0
  println(first)
  println(second)

  while (count < n){
    val sum = first + second
    first = second
    second = sum
    count = count + 1
    println(sum)
  }
  return first
}

fib1(10)
```

```
//o Write the function using recursion
def fib2(n:Int):Int= n match{
  case 0 | 1=> n
  case _ => fib2(n-1) + fib2(n-2)

}

fib2(20)
0 1 1 2 3 5 8 13 21 34 55 89 fib1: (n: Int)Int fib2: (n: Int)Int res4: Int =
6765
/*
Task 2
Create a calculator to work with rational numbers.
Requirements:
o It should provide capability to add, subtract, divide and multiply rational
numbers
o Create a method to compute GCD (this will come in handy during operations
on
rational)
Add option to work with whole numbers which are also rational numbers i.e.
(n/1)
- achieve the above using auxiliary constructors
- enable method overloading to enable each function to work with numbers and
rational.
*/
def calculator(n:Int,x:Int,y:Int):Int = n match{
  case 0 => return x + y
  case 1  => return x - y
  case 2 => return x * y
  case 3 => return x / y
  case _ => 0
}
calculator: (n: Int, x: Int, y: Int)Int
// subtract operation
```

```
calculator(1,3,4)
res5: Int = -1
//add operation
calculator(0,3,4)
res6: Int = 7
//multiply operation
calculator(2,3,4)
res7: Int = 12
//divide operation
calculator(3,12,4)
res9: Int = 3
//not a valid defined operation
calculator(4,4,4)
res10: Int = 0
//Create a method to compute GCD (this will come in handy during operations on rational)
def gcd(a: Double,b: Double): Double= {
if(b ==0.0) a else gcd(b, a%b)
}
gcd(20.0,15.35)
gcd: (a: Double, b: Double)Double res11: Double = 1.7763568394002505E-15
/*
Task 3
1.Write a simple program to show inheritance in scala.
2.Write a simple program to show multiple inheritance in scala.
*/

//1.Write a simple program to show inheritance in scala.

class Employee{
    var salary:Float = 10000
}

class Programmer extends Employee{
```

```
    var bonus:Int = 5000
    println("Salary = "+salary)
    println("Bonus = "+bonus)
}
val s = new Programmer()
```

```
Salary = 10000.0 Bonus = 5000 defined class Employee defined class Programmer
s: Programmer = Programmer@53f778ba
```

```
//2.Write a simple program to show multiple inheritance in scala.
class Employee{
    var salary:Float = 10000
}
class Designation extends Employee{
  var Desi1:String = "Architect"
  var Desi2:String = "Developer"
  var basci_arc:Float = 4345
  var basic_dev:Float = 2000


}
class Empdetails extends Designation{
  var org:String = "Technology Ltd."
  println("Oraganization Name : "+org)
  println("Desination : ")
  println("==========================================")
  println(Desi1)
  println(Desi2)
  println("==========================================")
  println("Salary Details ")
  println("==========================================")
  println("Salary of architect :"+salary)
  println("Basic of architect :"+basci_arc)
```

```
  println("Salary of developer :"+salary)
  println("Basic of developer :"+basic_dev)
  println("==========================================")
}
defined class Employee defined class Designation defined class Empdetails
//instantiating the object Empdetails
val s = new Empdetails()
Oraganization Name : Technology Ltd. Desination : ============================
================ Architect Developer ====================================
===== Salary Details ========================================== Salary of a
rchitect :10000.0 Basic of architect :4345.0 Salary of developer :10000.0 Bas
ic of developer :2000.0 ========================================== s: Empde
tails = Empdetails@21b37b58
/*
3.Write a partial function to add three numbers in which one number is
constant and two
numbers can be passed as inputs and define another method which can take the
partial
function as input and squares the result.


Partially Applied Functions
In functional programming languages, a call to a function that has parameters
can also be stated as applying the function to the parameters. When a
function is called with all the required parameters, it has fully applied the
function to all of the parameters. But when only a subset of the parameters
to the function is passed, the result of the expression is a Partially
Applied function. Scala does not throw an exception when you provide fewer
arguments to function, it simply applies them and returns a new function with
rest of arguments which need to be passed.



*/


val squareRoot: PartialFunction[Double, Double] = {
  case x if x >= 0 => Math.sqrt(x)
}
val addConstantTo: PartialFunction[(Int, Int), Int] = {
```

```
  case (a, b) => a + b + 12345
}

squareRoot: PartialFunction[Double,Double] = <function1> addConstantTo: Parti
alFunction[(Int, Int),Int] = <function1>
squareRoot(10)

res20: Double = 3.1622776601683795
addConstantTo(10,20)
res21: Int = 12375
/*

4.Write a program to print the prices of 4 courses of Acadgild: Android-
12999,Big Data
Development-17999,Big Data Development-17999,Spark-19999 using match and add
a
default condition if the user enters any other course
*/

def matchPrice(course: String): Unit =
  course match{
  case "Android" =>println("Price of Android course is rs 12999 only")
  case "Big Data Development" => println("Price of Big Data Development
course is rs 17999 only")
  case "Big Data Administration" => println("Price of Big Data Administration
course is rs 17999 only")
  case "spark" => println("Price of Spark course is rs 19999 only")
  case _ => println("No Course Found")
  }

matchPrice: (course: String)Unit
matchPrice("NLP")
No Course Found
matchPrice("Android")
Price of Android course is rs 12999 only
```

```
matchPrice("Big Data Development")
Price of Big Data Development course is rs 17999 only
matchPrice("Big Data Administration")
Price of Big Data Administration course is rs 17999 only
matchPrice("spark")
Price of Spark course is rs 19999 only
```