Task 1

```
/* A Fibonacci series (starting from 1) written in order without any spaces
in between, thus producing a sequence of digits.
This program finds the Nth digit in the sequence.
o Write the function using standard for loop
o Write the function using recursion
*/

//This function finds the Nth digit in the sequence (starting from 1) using
standard for loop
def fib_using_for_loop( n : Int ) : Int = {
  var a = 1
  var b = 1
  var i = 1

  for(i <- 2 to n) {
    val c = a + b
    a = b
    b = c
  }
  return a
}

//This function finds the Nth digit in the sequence (starting from 1) using
recursion
def fib_using_recursion( n : Int) : Int = n match {
   case 1 | 2 => 1
   case _ => fib_using_recursion( n-1 ) + fib_using_recursion( n-2 )
}

var n = 10

println("The "+ n +"th digit in the Fibonacci series using standard for
loop is: "+fib_using_for_loop(n)+"\n")
println("The "+ n +"th digit in the Fibonacci series using recursion is:
"+fib_using_recursion(n)+"\n")


The 10th digit in the Fibonacci series using standard for loop is: 55

The 10th digit in the Fibonacci series using recursion is: 55

fib_using_for_loop: (n: Int)Int
fib_using_recursion: (n: Int)Int
n: Int = 10
```

Task 2

```
/*This progam creates 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.
*/

//Defining Rational Class
import math._
class Rational(n: Int, d: Int) {
  //Creating a method gcd to compute GCD
  private def gcd(x: Int, y: Int): Int = {
    if (x == 0) y
    else if (x < 0) gcd(-x,y)
    else if (y < 0) -gcd(x, -y)
    else gcd(y % x, x)
  }
  private val g = gcd(n, d)
  val numer: Int = n/g
  val denom: Int = d/g
  def +(that: Rational) = new Rational(numer * that.denom + that.numer *
denom, denom * that.denom)
  def -(that: Rational) = new Rational(numer * that.denom - that.numer *
denom, denom * that.denom)
  def *(that: Rational) = new Rational(numer * that.numer, denom *
that.denom)
  def /(that: Rational) = new Rational(numer * that.denom, denom *
that.numer)
  override def toString = "" + numer + "/" + denom + ""
  def square = new Rational(numer*numer, denom*denom)
}

import math._
defined class Rational
```

```
//Defining Calculator Class
class Calculator(a: Rational, b: Rational, x: Double, y:Double){

  //Auxiliary Constructor for No Values
  def this() = this(new Rational(0,1),new Rational(0,1),0,0)

  //Auxiliary Constructor for Rational Values
  def this(a: Rational, b: Rational) = this(a,b,0,0)

  //Auxiliary Constructor for Double Values
  def this(x: Double, y: Double) = this(new Rational(0,1),new
Rational(0,1),x,y)

  //Defining add method for Int values
  def add(a: Rational, b: Rational):Rational = a + b

  //Defining subtract method for Int values
  def subtract(a: Rational, b: Rational):Rational = a - b

  //Defining divide method for Int values
  def divide(a: Rational, b: Rational):Rational = a / b

  //Defining multiply method for Int values
  def multiply(a: Rational, b: Rational): Rational = a * b

  //Overloading add method for Double Values
  def add(a: Double, b: Double ) = a + b

  //Overloading subtract method for Double Values
  def subtract(a: Double, b: Double ) = a - b

  //Overloading divide method for Double Values
  def divide(a: Double, b: Double )= a.toFloat/b

  //Overloading multiply method for Double Values
  def multiply(a: Double, b: Double ) = a * b
}

defined class Calculator

var a = new Rational(3,4)
var b = new Rational(10,3)

var p = 10
var q = 3

a: Rational = 3/4
b: Rational = 10/3
p: Int = 10
q: Int = 3
```

```
//Creating an object of Calculator class
var calc = new Calculator()

calc: Calculator = Calculator@6639476a

println("The addition of "+a+ " and "+b+ " is: "+calc.add(a, b)+"\n")
println("The substraction of "+a+ " and "+b+ " is: "+calc.subtract(a,
b)+"\n")
println("The division of "+a+ " and "+b+ " is: "+calc.divide(a, b)+"\n")
println("The multiplication of "+a+ " and "+b+ " is: "+calc.multiply(a,
b)+"\n")

println("The addition of "+p+ " and "+q+ " is: "+calc.add(p,q)+"\n")
println("The substraction of "+p+ " and "+q+ " is:
"+calc.subtract(p,q)+"\n")
println("The division of "+p+ " and "+q+ " is: "+calc.divide(p,q)+"\n")
println("The multiplication of "+p+ " and "+q+ " is:
"+calc.multiply(p,q)+"\n")

The addition of 3/4 and 10/3 is: 49/12

The substraction of 3/4 and 10/3 is: -31/12

The division of 3/4 and 10/3 is: 9/40

The multiplication of 3/4 and 10/3 is: 5/2

The addition of 10 and 3 is: 13.0

The substraction of 10 and 3 is: 7.0

The division of 10 and 3 is: 3.3333333333333335

The multiplication of 10 and 3 is: 30.0
```

Task 3

```
/*1. This program writes a simple program to show inheritance in scala.*/

//Defining base class: 'Person'
class Person{
  var AADHAR_card_id:String="9999-4455-6789"
}

/*
Student extends Person, it inherits the attribute holding the AADHAR card
id. In class Student, I print AADHAR_card_id and student_id.
I have defined a function student_information() which prints the student
detail i.e. AADHAR card id and student id.
*/
class Student extends Person{
  var student_id:String="1PI12IS002"

  println("AADHAR Card Id: "+AADHAR_card_id+"\n")
  println("Student Id: "+student_id+"\n")

  def student_information(){
    println("Student with AADHAR card id: "+AADHAR_card_id+" is enrolled
with student id: "+ student_id+".\n")
  }
}

defined class Person
defined class Student

//Createing an object of class Student.
var stud = new Student()


AADHAR Card Id: 9999-4455-6789

Student Id: 1PI12IS002

stud: Student = Student@5b2acdde

//Calling the student_information() function defined in Student class
stud.student_information()

Student with AADHAR card id: 9999-4455-6789 is enrolled with student id: 1
PI12IS002.
```

Task 3

```
/*2. This program writes a simple program to show multiple inheritance in
scala.*/

//Creating the base abstract class: 'Person'
abstract class Person{
  def GetUserId:String
}

/*
Now, creating two traits each one overriding GetUserId function with one
additional function specific to the subclass
*/

//Creating Student trait which extends the abstract class 'Person' and
overridden GetUserId method and added one GetCurrentCGPA method
trait Student extends Person{
  override def GetUserId = "1PI12IS002"
  def GetCurrentCGPA()= "9.6"
}

//Creating Professional trait which extends the abstract class 'Person' and
overridden GetUserId method and added one GetCurrentSalary method
trait Professional extends Person{
  override def GetUserId = "MSFT10095"
  def GetCurrentSalary() = "24 LPA"
}

/*
class StudentProfessional extends Student and Professional, it inherits the
methods GetUserId, GetCurrentCGPA and GetCurrentSalary.
I have defined a method GetStudentInformation() which prints the student
professional detail i.e. user id, CGPA and salary.
In StudentProfessional class, as “with Professional” is written, so common
method(i.e. GetUserId) is taken from  Professional class hence I get
GetUserId value from Professional class.
*/

class StudentProfessional extends Student with Professional{

  println("Student Professional user id is: "+GetUserId+".\n")
  println("Student Professionl current CGPA is: "+GetCurrentCGPA+".\n")
  println("Student Professional current salary is:
"+GetCurrentSalary+".\n")

  def GetStudentInformation(){
    println("Student professional with user id: "+GetUserId+ " has current
CGPA: "+GetCurrentCGPA+ ", is getting "+GetCurrentSalary+" as salary.\n")
  }
}
```

```
Student Professional user id is: MSFT10095.

Student Professionl current CGPA is: 9.6.

Student Professional current salary is: 24 LPA.

Student professional with user id: MSFT10095 has current CGPA: 9.6, is get
ting 24 LPA as salary.

defined class Person
defined trait Student
defined trait Professional
defined class StudentProfessional
sp: StudentProfessional = StudentProfessional@3c4ef8df

//Createing an object of StudentProfessional class.
var sp =  new StudentProfessional()


Student Professional user id is: MSFT10095.

Student Professionl current CGPA is: 9.6.

Student Professional current salary is: 24 LPA.

sp: StudentProfessional = StudentProfessional@4510f552

//Calling the get_student_information() function defined in
StudentProfessional class.
sp.GetStudentInformation()

Student professional with user id: MSFT10095 has current CGPA: 9.6, is get
ting 24 LPA as salary.
```

Task 3

```
/*3. This program writes 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.*/

//Defining add_three_num partial function which adds three numbers in which
one number is constant and two numbers can be passed as inputs
val add_three_num: PartialFunction[(Int,Int), Int] = {
 case(x,y) => x+y+9
}

//Defining calc_square function which takes an Int variable as input and
returns square of the input variable.
def calc_square(x: Int): Int = x * x

//Defining calc_func_square method which takes add_three_num partial
function as input and squares the result
def cal_func_square(add_three_num:Int):Int =(calc_square(add_three_num))

var x = 4
var y = 5

println("The sum of "+x+" , "+y+" and 9 is: "+add_three_num(x,y)+".\n")
println("The square of "+add_three_num(x,y)+" is:
"+cal_func_square(add_three_num(x,y))+".\n")

The sum of 4 , 5 and 9 is: 18.

The square of 18 is: 324.

add_three_num: PartialFunction[(Int, Int),Int] = <function1>
calc_square: (x: Int)Int
cal_func_square: (add_three_num: Int)Int
x: Int = 4
y: Int = 5

The price of Android course @ AcadGild is: Rs. 12999\-.

The price of Big Data Development course @ AcadGild is: Rs. 17999\-.

The price of Spark course @ AcadGild is: Rs. 19999\-.

The price of Cloud Computing Advanced course @ AcadGild is: Rs. Currently,
we are not offering this course. Please try other courses.

acadgild_course_price: (course_name: String)String
```