databricks Assignment_22.2_All_Solutions

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 = 0
  var b = 1
  var i = 1

  for(i <- 1 to n+1) {
    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 => n
   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: 89

The 10th digit in the Fibonacci series using recursion is: 89

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 Calculator Class
class Calculator(a: Int, b: Int, x: Double, y:Double){

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

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

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

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

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

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

  //Defining gcd method for Int value
  def gcd(a:Int, b:Int): Int = {
       if(b ==0) a else gcd(b, 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 / b
  //Overloading multiply method for Double Values
  def multiply(a: Double, b: Double ) = a * b

  //Overloading gcd method for Double Values
  def gcd(a: Double,b: Double): Double = {
       if(b ==0) a else gcd(b, a%b)
    }
}

var a = 4
var b = 5

//Creating an object of Calculator class
var calc = new Calculator(a, b)

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

println("The addition of "+a+ " and "+b+ " is:
"+calc.add(a.asInstanceOf[Double],b.asInstanceOf[Double])+".\n")
println("The substraction of "+a+ " and "+b+ " is:
"+calc.subtract(a.asInstanceOf[Double],b.asInstanceOf[Double])+".\n")
println("The division of "+a+ " and "+b+ " is:
"+calc.divide(a.asInstanceOf[Double],b.asInstanceOf[Double])+".\n")
println("The multiplication of "+a+ " and "+b+ " is:
"+calc.multiply(a.asInstanceOf[Double],b.asInstanceOf[Double])+".\n")
println("The GCD of "+a+ " and "+b+ " is:
"+calc.gcd(a.asInstanceOf[Double],b.asInstanceOf[Double])+".\n")
```

```
The addition of 4 and 5 is: 9.

The substraction of 4 and 5 is: -1.

The division of 4 and 5 is: 0.

The multiplication of 4 and 5 is: 20.

The GCD of 4 and 5 is: 1.

The addition of 4 and 5 is: 9.0.

The substraction of 4 and 5 is: -1.0.
The division of 4 and 5 is: 0.8.

The multiplication of 4 and 5 is: 20.0.

The GCD of 4 and 5 is: 1.0.

defined class Calculator
a: Int = 4
b: Int = 5
calc: Calculator = Calculator@64d6a336
```

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")
  }
}

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

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

```
AADHAR Card Id: 9999-4455-6789

Student Id: 1PI12IS002

Student with AADHAR card id: 9999-4455-6789 is enrolled with student id: 1PI12I
S002.

defined class Person
defined class Student
stud: Student = Student@1e1f3e43
```

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")
  }
}

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

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

```
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 getting
24 LPA as salary.

defined class Person
defined trait Student
defined trait Professional
defined class StudentProfessional
sp: StudentProfessional = StudentProfessional@28b1dcb4
```

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 function which adds three numbers in which one number
is constant and two numbers can be passed as inputs
def add_three_num(x: Int, y:Int, c:Int =9) = x + y + c

//Defining calc_square function which takes the partial function as input and
squares the result.
def calc_square(x: Int): Int = x * x

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:
"+calc_square(add_three_num(4,5))+".\n")
```

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

The square of 18 is: 324.

add_three_num: (x: Int, y: Int, c: Int)Int
calc_square: (x: Int)Int
x: Int = 4
y: Int = 5

The price of Android-12999 course @ AcadGild is: Rs. 30,000/-.

The price of Big Data Development-17999 course @ AcadGild is: Rs. 60,000/-.

The price of Data Science Masters-11439 course @ AcadGild is: Rs. 60,180/-.

The price of Spark-19999 course @ AcadGild is Rs. 90,000/-.

The price of Cloud Computing Advanced-134567 course @ AcadGild is: Currently, w
e are not offering this course. Please try other courses.

acadgild_course_price: (course_name: String)String
```