Task 1

Write a simple program to show inheritance in scala.

```
scala> class Employee{
     | var salary: Float = 10000
     | }
defined class Employee

scala> class Programmer extends Employee{
     | var bonus: Int = 5000
     | println("salary = "+ salary)
     | println("bonus = " + bonus)
     | }
defined class Programmer

scala> object MainObject{
     | def main(args:Array[String]){
     | new Programmer()
     | }
     | }
defined object MainObject

scala> MainObject
res20: MainObject.type = MainObject$@32ccaba7

scala> var programmer1 = new Programmer()
salary = 10000.0
bonus = 5000
programmer1: Programmer = Programmer@393d64d2

scala>
```

Task 2

Write a simple program to show multiple inheritance in scala

```
scala> trait A {
     |          var distance: Int = _
     |          def action = {
     |            distance = distance + 5
     |          }
     |        }
defined trait A

scala>
     |        trait B {
     |          var driverVar: Int = _
     |          def action = {
     |            driverVar = driverVar + 1
     |          }
     |        }
defined trait B

scala>
     |        class AB extends A with B {
     |         distance = 3;
     |         driverVar = 6;
     |         override def action = {
     |          super[A].action
     |          super[B].action
     |         }
     |        }
defined class AB

scala> var ab = new AB
ab: AB = AB@281ca256

scala> ab.action

scala> print(ab.driverVar)
7
scala> print(ab.distance)
8
```

Task 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.

```
scala> val valueConstant = 10;
valueConstant: Int = 10

scala> var partFunction: PartialFunction[(Int,Int),Int] = {
     | case(x,y) => x + y + valueConstant
     | }
partFunction: PartialFunction[(Int, Int),Int] = <function1>

scala> partFunction(10,10)
res0: Int = 30

scala> def square(f:Int) = f * f
square: (f: Int)Int

scala> square(partFunction(10,10))
res1: Int = 900

scala>
```

Task 4

Write a program to print the prices of 4 courses of Acadgild:

Android App Development -14,999 INR
Data Science - 49,999 INR
Big Data Hadoop & Spark Developer - 24,999 INR

Blockchain Certification - 49,999 INR

using match and add a default condition if the user enters any other course.

```
scala> case class AcadgildCourses(courseName: String)
defined class AcadgildCourses

scala> var android = AcadgildCourses("Android")
android: AcadgildCourses = AcadgildCourses(Android)

scala> var dataScience = AcadgildCourses("DataScience")
dataScience: AcadgildCourses = AcadgildCourses(DataScience)

scala> var bigDataAndSpark = AcadgildCourses("BigData")
bigDataAndSpark: AcadgildCourses = AcadgildCourses(BigData)

scala> var blockchainCertification = AcadgildCourses("BlockChain")
blockchainCertification: AcadgildCourses = AcadgildCourses(BlockChain)

scala> var others = AcadgildCourses("Java")
others: AcadgildCourses = AcadgildCourses(Java)

scala> for (courses <- List(android, dataScience, bigDataAndSpark, blockchainCertification, others)) {
     |           courses match {
     |              case AcadgildCourses("Android") => println("Android App Development, 14999 INR")
     |              case AcadgildCourses("DataScience") => println("Data Science, 49990 INR")
     |              case AcadgildCourses("BigData") => println("Big Data Hadoop and Spark Developer, 24000 INR")
     |       case AcadgildCourses("BlockChain") => println("Blockchain Certification, 49000 INR")
     |       case AcadgildCourses(courseName) => println("Others Course  ")
     |          }
     |       }
Android App Development, 14999 INR
Data Science, 49990 INR
Big Data Hadoop and Spark Developer, 24000 INR
Blockchain Certification, 49000 INR
Others Course

scala>
```