****

**Course Title: Programming Language II**

**Course Code: CSE 111**

**Lab Assignment no: 7**

*\*\* You are not allowed to change any of the code of the tasks*

*\*\* Use* ***Inheritance*** *to solve all problems*

Task - 1

Given the following classes, write the code for the **BBA\_Student** class so that the following output is printed:

| class Student:  def \_\_init\_\_(self, name='Just a student', dept='nothing'):  self.\_\_name = name  self.\_\_department = dept  def set\_department(self, dept):  self.\_\_department = dept  def get\_name(self):  return self.\_\_name  def set\_name(self,name):  self.\_\_name = name  def \_\_str\_\_(self):  return 'Name: '+self.\_\_name+' Department: '+self.\_\_department  #write your code here  print(BBA\_Student()) print(BBA\_Student('Humpty Dumpty')) print(BBA\_Student('Little Bo Peep')) | *Output:*  Name: default Department: BBA  Name: Humpty Dumpty Department: BBA  Name: Little Bo Peep Department: BBA |
| --- | --- |

Task – 2

| class Vehicle:  def \_\_init\_\_(self):  self.x = 0  self.y = 0  def moveUp(self):  self.y+=1  def moveDown(self):  self.y-=1  def moveRight(self):  self.x+=1  def moveLeft(self):  self.x-=1  def \_\_str\_\_(self):  return '('+str(self.x)+' , '+str(self.y)+')'  #write your code here  print('Part 1') print('------') car = Vehicle() print(car) car.moveUp() print(car) car.moveLeft() print(car) car.moveDown() print(car) car.moveRight() print(car) print('------') print('Part 2') print('------') car1 = Vehicle2010() print(car1) car1.moveLowerLeft() print(car1) car2 = Vehicle2010() car2.moveLeft() print(car1.equals(car2)) car2.moveDown() print(car1.equals(car2)) | *OUTPUT:*  Part 1  ------  (0 , 0)  (0 , 1)  (-1 , 1)  (-1 , 0)  (0 , 0)  ------  Part 2  ------  (0 , 0)  (-1 , -1)  False  True |
| --- | --- |

A vehicle assumes that the whole world is a 2-dimensional graph paper. It maintains its x and y coordinates (both are integers). The vehicle gets manufactured (constructed) at (0, 0) coordinate.

Subtasks:

1. Design a **Vehicle2010 class** which inherits movement methods from **Vehicle** and adds new methods called **move UpperRight, UpperLeft, LowerRight, LowerLeft.** Each of these diagonal move methods must re-use two inherited and appropriate move methods.
2. Write an “**equals**” method which tests if significant class properties are the same (in this case x and y).

**Note: All moves are 1 step. That means a single call to any move method changes value of either x or y or both by 1.**

Task - 3

Given the following classes, write the code for the **Cricket\_Tournament** and the **Tennis\_Tournment** class so that the following output is printed.

| class Tournament:      def \_\_init\_\_(self,name='Default'):          self.\_\_name = name      def set\_name(self,name):          self.\_\_name = name      def get\_name(self):          return self.\_\_name  #write your code here  ct1 = Cricket\_Tournament()  print(ct1.detail())  print("-----------------------")  ct2 = Cricket\_Tournament("IPL",10,"t20")  print(ct2.detail())  print("-----------------------")  tt = Tennis\_Tournament("Roland Garros",128)  print(tt.detail()) | *OUTPUT:*  Cricket Tournament Name: Default Number of Teams: 0  Type: No type  -----------------------  Cricket Tournament Name: IPL  Number of Teams: 10  Type: t20  -----------------------  Tennis Tournament Name: Roland Garros  Number of Players: 128 |
| --- | --- |

Task - 4

Given the following classes, write the code for the **Book** and the **CD** class so that the following output is printed.

| class Product:      def \_\_init\_\_(self,id, title, price):          self.\_\_id = id          self.\_\_title = title          self.\_\_price = price      def get\_id\_title\_price(self):  return "ID: "+str(self.\_\_id)+" Title:"+self.\_\_title+ "Price: "+str(self.\_\_price)  #write your code here  book = Book(1,"The Alchemist",500,"97806","HarperCollins")  print(book.printDetail())  print("-----------------------")  cd = CD(2,"Shotto",300,"Warfaze",50,"Hard Rock")  print(cd.printDetail()) | *OUTPUT:*  ID: 1 Title: The Alchemist Price: 500 ISBN: 97806 Publisher: HarperCollins  -----------------------  ID: 2 Title: Shotto Price: 300  Band: Warfaze Duration: 50 minutes Genre: Hard Rock |
| --- | --- |

Task - 5

Given the following classes, write the code for the **Dog** and the **Cat** class so that the following output is printed.

| class Animal:   def \_\_init\_\_(self,sound):  self.\_\_sound = sound    def makeSound(self):  return self.\_\_sound    class Printer:   def printSound(self, a):  print(a.makeSound())  #write your code here  d1 = Dog('bark') c1 = Cat('meow') a1 = Animal('Animal does not make sound')  pr = Printer() pr.printSound(a1) pr.printSound(c1) pr.printSound(d1) | *OUTPUT:*  Animal does not make sound  meow  bark |
| --- | --- |

Task - 6

Given the following classes, write the code for the **Triangle** and the **Trapezoid** class so that the following output is printed.

| class Shape:    def \_\_init\_\_(self, name='Default', height=0, base=0):      self.area = 0      self.name = name      self.height = height      self.base = base    def get\_height\_base(self):      return "Height: "+str(self.height)+",Base: "+str(self.base)  #write your code here  tri\_default = triangle()  tri\_default.calcArea()  print(tri\_default.printDetail())  print('--------------------------')  tri = triangle('Triangle', 10, 5)  tri.calcArea()  print(tri.printDetail())  print('---------------------------')  trap = trapezoid('Trapezoid', 10, 6, 4)  trap.calcArea()  print(trap.printDetail()) | *OUTPUT:*  Shape name: Default  Height: 0, Base: 0  Area: 0.0  ---------------------------  Shape name: Triangle  Height: 10, Base: 5  Area: 25.0  ---------------------------  Shape name: Trapezoid  Height: 10, Base: 6, Side\_A: 4  Area: 50.0 |
| --- | --- |

Task - 7

Given the following classes, write the code for the **Player** and the **Manager** class so that the following output is printed. To calculate the match earning use the following formula:

1. Player: (total\_goal \* 1000) + (total\_match \* 10)
2. Manager: match\_win \* 1000

| class Football:    def \_\_init\_\_(self, team\_name, name, role):      self.\_\_team = team\_name      self.\_\_name = name      self.role = role      self.earning\_per\_match = 0    def get\_name\_team(self):      return 'Name: '+self.\_\_name+', Team Name: ' +self.\_\_team  #write your code here  player\_one = Player('Juventus', 'Ronaldo', 'Striker', 25, 32)  player\_one.calculate\_ratio()  player\_one.print\_details()  print('------------------------------------------')  manager\_one = Manager('Real Madrid', 'Zidane', 'Manager', 25)  manager\_one.print\_details() | *OUTPUT:*  Name: Ronaldo, Team Name: Juventus  Team Role: Striker  Total Goal: 25, Total Played: 32  Goal Ratio: 0.78125  Match Earning: 25320K  ----------------------------------  Name: Zidane, Team Name: Real Madrid  Team Role: Manager  Total Win: 25  Match Earning: 25000K |
| --- | --- |