通識計算機程式設計期末考參考解答

6/29/2018

1. 。

(a) 撰寫介面 **TextReader**，宣告的函式為 **FileName**、**Open**、**Close**、**Read**。這四個函式，呼叫時都不需要參數，但 **FileName** 要傳回字串，**Read** 要傳回字串陣列。 (3%)

Ans.

```
  interface TextReader  {
   string FileName();
   void Open();
   void Close();
   string[] Read();
   }
```

(b) 撰寫實作 **TextReader** 之 **WindowsTextReader** **類別**建構式：設定成員變數 **fileName** 等於建構式參數（圖 2 例中為字串"**wtr**"），並且在主控台螢幕顯示如圖 2 之第一行訊息。 (6%)

Ans.

```
   public WindowsTextReader(string fileName)
   {
       this.fileName = fileName;
       Console.WriteLine("Windows Text Reader: set file to be " +
           fileName);
    }
```

(c) 撰寫 **WindowsTextReader** 中的成員函式 **Open** 及 **GetData**。函式 **Open** 先在主控台顯示一行訊息，內容如圖 2 第 2 行。其次呼叫成員函式 **GetData**。函式 **GetData** 將檔案內容移入記憶緩衝器 **buffer**。這裡簡化問題，假設 Windows 檔案內容等於字串陣列 **{"Hello!", "World."}**，直接於 **GetData**，將此陣列設值給 **buffer** 即可。**GetData** 也要顯示如圖 2 第三行的訊息。(6%)

Ans.

```
public void Open() {
    Console.WriteLine("Windows Text Reader: Open file " +
      fileName);
    buffer = GetData();
 }
 private string[] GetData()  {
    Console.WriteLine("Windows Text Reader: Get data");
    string[] contents = { "Hello!", "World." };
    return contents;
 }
```

(d) 建立類別 **WindowsTextReader** 中的成員函式 **Read**，先顯示訊息如圖 2 第 7 行，再傳回 **buffer**。 (6%)

Ans.

```
public string[] Read()
 {
    Console.WriteLine(
      "Windows Text Reader: Read from buffer ");
    return buffer;
 }
```

(e) 寫類別 **Program** 中的函式 **CreateAndOpen** 及 **ReadAndDisplay**。函式 **CreateAndOpen** 先建立 **TextReader** 陣列。第一個元素是 **WindowsTextReader** 物件，對應的檔案名稱是**"wtr"**，第二個元素是 **iOSTextReader** 物件，對應的檔案名稱是**"atr"**。接著呼叫這兩個元素物件的成員函式 **Open**。至於函式 **ReadAndDisplay**，則依次呼叫兩個陣列元素物件的 **Read** 函式，再以 **Read** 函式傳回的字串陣列，呼叫如下函式 **Display**。(6%)

```
 static void Display(string[] contents) {
    int nTerms = contents.Length;
    for(int i = 0; i < nTerms; ++i)  {
       Console.Write(contents[i] + "  ");
    }
    Console.WriteLine();
 }
```

Ans.

```
static TextReader[] CreateAndOpen()
{
    readers[0] = new WindowsTextReader("wtr");
    readers[0].Open();
    readers[1] = new iOSTextReader("atr");
    readers[1].Open();
   return readers;
}

static void ReadAndDisplay(TextReader[] readers)
{
    string[] contents;
    int nReaders = readers.Length;
    for (int i = 0; i < nReaders; ++i)
    {
        contents = readers[i].Read();
        Display(contents);
        readers[i].Close();
    }
}
```

(f) 完成類別 **Program** 之主程式 **Main**。假定 **WindowsTextReader** 類別的所有成員函式均已完成。而 **iOSTextReader** 類別，也已經以類似 **WindowsTextReader** 類別的方式，實作完畢，可以直接引用。(3%)

Ans.

```
static void Main(string[] args)
{
    TextReader[] readers;
    CreateAndOpen(out readers);
    ReadAndDisplay(readers);
}
```

2. 找出以下程式片段之錯誤，並予更正.

(a) (3%)

Ans. 兩個 Magnitude 函式僅有傳回值的型別不同,
不合多載(overloading)的要求.
其中一個 Magnitude 函式需要改名

```
class Vector2D {
    private double x;
    private double y;
    public Vector2D(double x, double y)  {
        this.x = x;
        this.y = y;
    }
    public double ~~Magnitude~~ DMagnitude()  {
        return Math.Sqrt(x * x + y * y);
    }
    public float ~~Magnitude~~ FMagnitude()  {
        return (float)Math.Sqrt(x * x + y * y);
    }
}
```

(b) (3%) 一個錯誤

Ans.

子類別的PayBill函式，用到了父類別Parents的private成員變數cash，不合乎繼承的規定，應該將 cash 改為 protected。

```
class Parents  {
    private string dad;
    private string mom;
    private int pension;
    ~~private~~ protected int cash;
    public Parents(string dad, string mom, int pension)  {
        this.dad = dad;
        this.mom = mom;
        this.pension = pension;
        cash = pension;
    }
    virtual public void PayBill(int bill)  {
        cash -= bill;
    }
}

class Child : Parents  {
    private string name;
    private int income;
    private int balance;
    public Child(string dad, string mom, int pension,
        string name, int income) : base(dad, mom, pension)  {
        this.name = name;
        this.income = income;
        balance = income;
    }
    override public void PayBill(int bill)  {
        balance -= bill;
        if(balance < 0)  {
            balance += cash;
        }
    }
}
```

(c) (3%) 一個錯誤。

Ans.

Airplane的成員函式 Fly 宣告為 abstract，使 Airplane 成為一個抽象類別，類別名稱應該加上 abstract 字樣。

```
abstract class Airplane  {
    abstract public void Fly();
}
class Boeing747_300 : Airplane  {
    override public void Fly()  {
        Console.WriteLine("Carry 300+ travellers.");
    }
 }
```

(d) (3%) 一種錯誤

抽象類別不能多重繼承，必須改成 **interface**

```
~~abstract class~~ interface Radio  {
    ~~abstract public~~ void PlayMusic();
}
~~abstract class~~ interface Clock  {
    ~~abstract public~~ void SetAlarm();
}
class RadioClock : Radio, Clock  {
    public ~~override~~ void SetAlarm()  {
        Console.WriteLine("Alarm is set");
    }
    public ~~override~~ void PlayMusic() {
        Console.WriteLine("Lullaby is played");
    }
}
```

(e) (3%) 一種錯誤

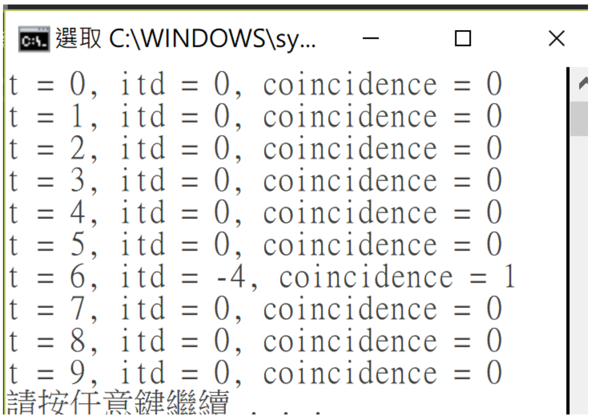
Ans.

物件產生的個數 **nCardsProduced** 與物件無關，應該宣告為 **static**，對應的函式也要改成 **static**。

```
 class Car {
    static private int nCarsProduced = 0;
    public Car() {
        Console.WriteLine("A car is produced in Detroit");
        ++nCarsProduced;
    }
    public Car(string city)  {
        Console.WriteLine("A car is produced in " + city);
        ++nCarsProduced;
    }
    static public int Total()  {
        return nCarsProduced;
    }
}
```

```
class Program  {
    static void Main(string[] args) {
        Car ford = new Car();
        Car toyota = new Car("Aichi-ken, Japan");
        Car luxgen = new Car("Miaoli county, Taiwan");
        Console.WriteLine("Total number of cars produced = {0}",
            Car.Total());
    }
}
```

3. 試寫出下列程式的輸出 (12%)

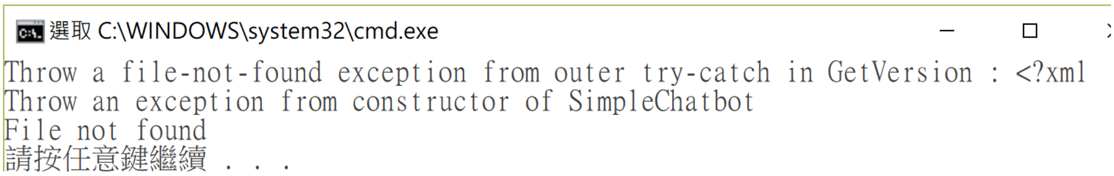


```
t = 0, itd = 0, coincidence = 0
t = 1, itd = 0, coincidence = 0
t = 2, itd = 0, coincidence = 0
t = 3, itd = 0, coincidence = 0
t = 4, itd = 0, coincidence = 0
t = 5, itd = 0, coincidence = 0
t = 6, itd = -4, coincidence = 1
t = 7, itd = 0, coincidence = 0
t = 8, itd = 0, coincidence = 0
t = 9, itd = 0, coincidence = 0
請按任意鍵繼續 . . .
```

4. 試寫出以下程式在下列狀況時的主控台螢幕輸出。

(a) (3%) 檔案 **test.aiml** 尚未建立。

Ans.

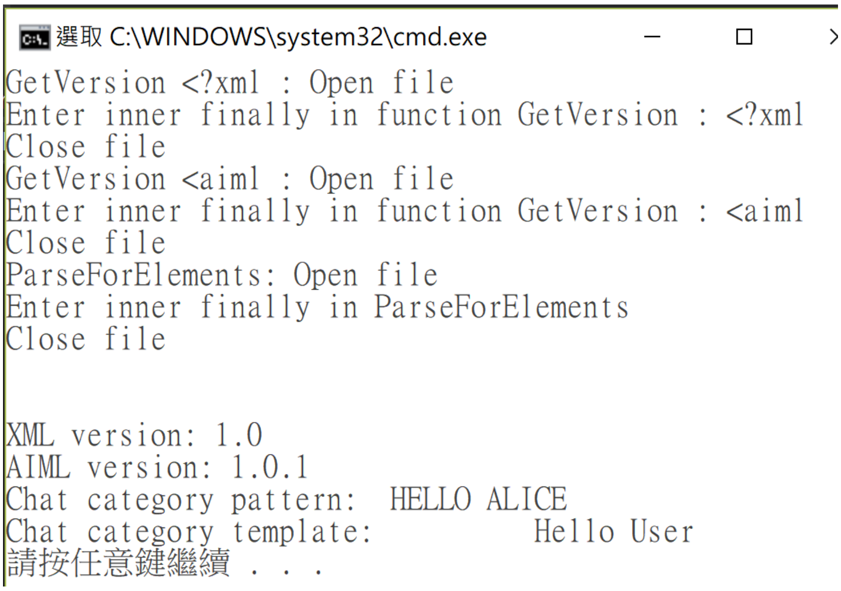


```
Throw a file-not-found exception from outer try-catch in GetVersion : <?xml
Throw an exception from constructor of SimpleChatbot
File not found
請按任意鍵繼續 . . .
```

(b) (3%) 檔案 **test.aiml** 已在正確位置，且內容為

```
<?xml version = "1.0" encoding = "UTF-8"?>
<aiml version="1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern> HELLO ALICE </pattern>
    <template>
      Hello User
    </template>
  </category>
</aiml>
```

Ans.



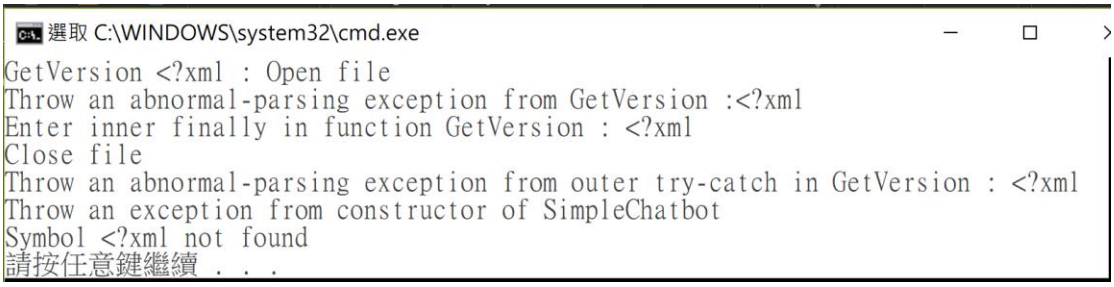
```
GetVersion <?xml : Open file
Enter inner finally in function GetVersion : <?xml
Close file
GetVersion <aiml : Open file
Enter inner finally in function GetVersion : <aiml
Close file
ParseForElements: Open file
Enter inner finally in ParseForElements
Close file


XML version: 1.0
AIML version: 1.0.1
Chat category pattern:  HELLO ALICE
Chat category template:         Hello User
請按任意鍵繼續 . . .
```

(c) (3%) 檔案 **test.aiml** 已在正確位置，且內容為

```
<?xml version = "1.0" encoding = "UTF-8"?>
  <category>
    <pattern> HELLO ALICE </pattern>
    <template>
      Hello User
    </template>
  </category>
</aiml>
```

Ans.

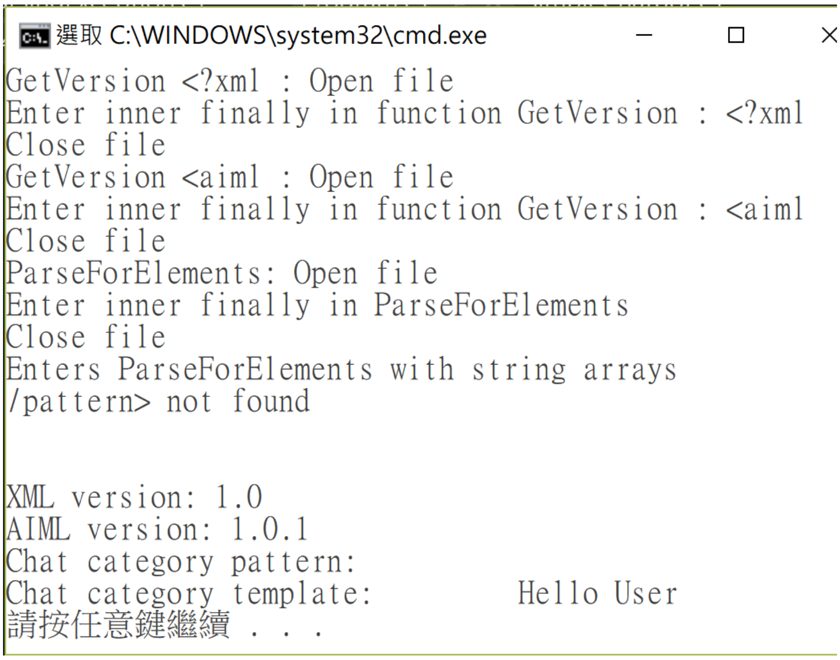


```
GetVersion <?xml : Open file
Throw an abnormal-parsing exception from GetVersion :<?xml
Enter inner finally in function GetVersion : <?xml
Close file
Throw an abnormal-parsing exception from outer try-catch in GetVersion : <?xml
Throw an exception from constructor of SimpleChatbot
Symbol <?xml not found
請按任意鍵繼續 . . .
```

(d) (3%) 檔案 **test.aiml** 已在正確位置，且內容為

```
<?xml version = "1.0" encoding = "UTF-8"?>
<aiml version="1.0.1" encoding = "UTF-8"?>
   <category>
      <pattern> HELLO ALICE
      <template>
         Hello User
      </template>
   </category>
</aiml>
```

Ans.

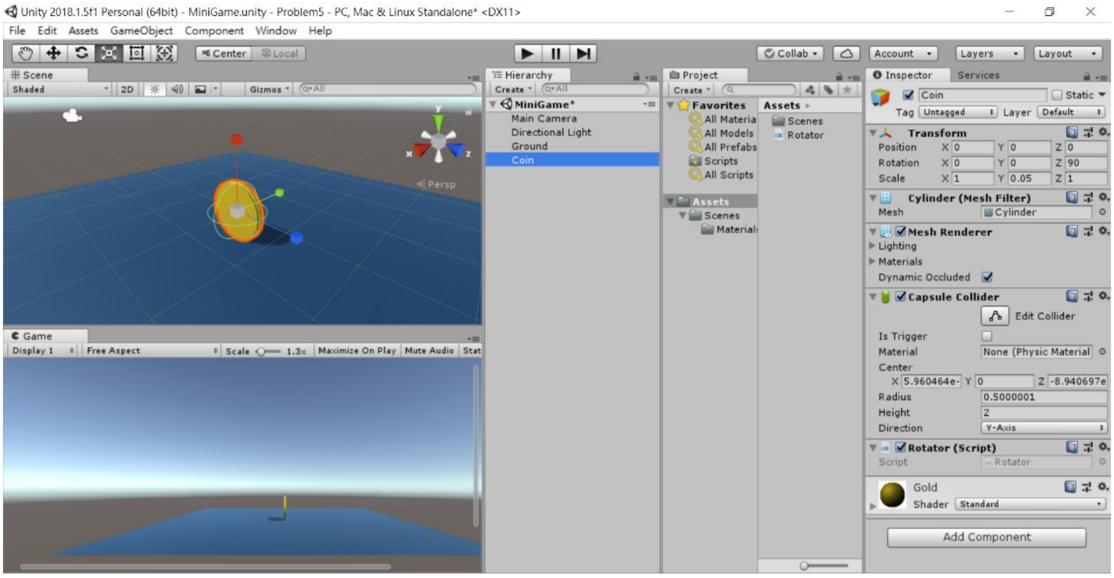


```
GetVersion <?xml : Open file
Enter inner finally in function GetVersion : <?xml
Close file
GetVersion <aiml : Open file
Enter inner finally in function GetVersion : <aiml
Close file
ParseForElements: Open file
Enter inner finally in ParseForElements
Close file
Enters ParseForElements with string arrays
/pattern> not found


XML version: 1.0
AIML version: 1.0.1
Chat category pattern:
Chat category template:         Hello User
請按任意鍵繼續 . . .
```

5. 依據以下描述及程式框架，完成 C#之 Unity 腳本程式。 (6%)

程式描述：建立繞全域坐標系 y 軸不停旋轉之金幣。
利用 Unity 使用介面，完成金幣外形與場景設計，如圖 3 。



Coin ☑ □ Static
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 90
Scale X 1 Y 0.05 Z 1

Cylinder (Mesh Filter)
Mesh Cylinder

☑ Mesh Renderer
Lighting
Materials
Dynamic Occluded ☑

☑ Capsule Collider
Edit Collider
Is Trigger □
Material None (Physic Material
Center
X 5.960464e- Y 0 Z -8.940697e
Radius 0.5000001
Height 2
Direction Y-Axis

☑ Rotator (Script)
Script Rotator

Gold
Shader Standard

圖 3 . 旋轉金幣之外形與場景設計

金幣旋轉的 C#腳本程式敘述為:

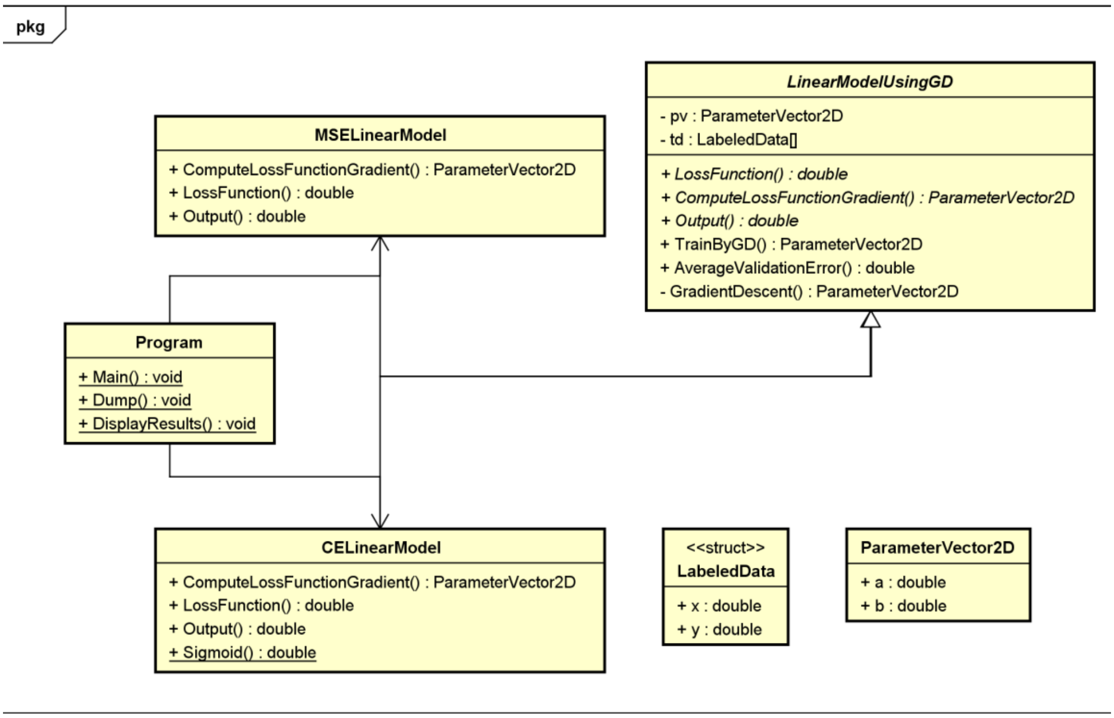
```
transform.Rotate(new Vector3(50, 0, 0) * Time.deltaTime);
```

請指出這一行敘述，應該放在以下程式框架中的哪一個位置? 這是單選題，回答選項 A、B、C、D、E 之一即可。

```
Ans. E
    void Update () {
       //**********************************
       //(E)
          transform.Rotate(new Vector3(50, 0, 0) * Time.deltaTime);
       //**********************************
    }
}
```

6.

參考解答類別圖


pkg
LinearModelUsingGD
- pv : ParameterVector2D
- td : LabeledData[]
+ LossFunction() : double
+ ComputeLossFunctionGradient() : ParameterVector2D
+ Output() : double
+ TrainByGD() : ParameterVector2D
+ AverageValidationError() : double
- GradientDescent() : ParameterVector2D
MSELinearModel
+ ComputeLossFunctionGradient() : ParameterVector2D
+ LossFunction() : double
+ Output() : double
Program
+ Main() : void
+ Dump() : void
+ DisplayResults() : void
CELinearModel
+ ComputeLossFunctionGradient() : ParameterVector2D
+ LossFunction() : double
+ Output() : double
+ Sigmoid() : double
<<struct>>
LabeledData
+ x : double
+ y : double
ParameterVector2D
+ a : double
+ b : double

```
// Problem 6
using System;

namespace Problem6
{
    class Program
    {
        static void Main(string[] args)
        {
            int nEpochs = 10;
            double gamma;
            ParameterVector2D pv_hat;
            double averageValidationError;

            // mean square loss model
            Console.WriteLine("Mean Square Error Loss Model  \n");
            LabeledData[] trainingDataForRegression =
            {
                new LabeledData(-5.00, -1.87),
                new LabeledData(-4.00, -1.52),
                new LabeledData(-3.00, -1.10),
                new LabeledData(-2.00, -0.71),
                new LabeledData(-1.00, -0.29),
                new LabeledData(0.00, 0.08),
                new LabeledData(1.00, 0.53),
                new LabeledData(2.00, 0.86),
                new LabeledData(3.00, 1.32),
                new LabeledData(4.00, 1.67)
            };
            Console.WriteLine("Training data");
            Dump(trainingDataForRegression);
            Console.WriteLine();

            // generating validation data
            LabeledData[] validationDataForRegression =
            {
                new LabeledData( 7.90,  3.26),
                new LabeledData( 7.64,  3.16),
                new LabeledData(-2.45, -0.88),
                new LabeledData(-0.75, -0.20),
                new LabeledData( 2.40,  1.06)
            };
            Console.WriteLine("Validata data");
            Dump(validationDataForRegression);
            Console.WriteLine();

            MSELinearModel mseModel =
                new MSELinearModel(trainingDataForRegression);
            gamma = 0.05;
            pv_hat = mseModel.TrainByGD(nEpochs, gamma);
            averageValidationError =
                mseModel.AverageValidationError(
                   validationDataForRegression);
            DisplayResults(nEpochs, pv_hat, averageValidationError);
            Console.WriteLine(
                "==============================================");
```

```
// cross entropy model
Console.WriteLine("Cross Entropy Model  \n");

LabeledData[] trainingDataForClassification =
{
   new LabeledData(-5.00, 0.00),
   new LabeledData(-4.00, 0.00),
   new LabeledData(-3.00, 0.00),
   new LabeledData(-2.00, 0.00),
   new LabeledData(-1.00, 0.00),
   new LabeledData( 0.00, 1.00),
   new LabeledData( 1.00, 1.00),
   new LabeledData( 2.00, 1.00),
   new LabeledData( 3.00, 1.00),
   new LabeledData( 4.00, 1.00)
};
Console.WriteLine("Training data");
Dump(trainingDataForClassification);
Console.WriteLine();

// generating validation data
LabeledData[] validationDataForClassification =
{
   new LabeledData( 7.90, 1.00),
   new LabeledData( 7.64, 1.00),
   new LabeledData(-2.45, 0.00),
   new LabeledData(-0.75, 0.00),
   new LabeledData( 2.40, 1.00)
};
Console.WriteLine("Validation data");
Dump(validationDataForClassification);
Console.WriteLine();

CELinearModel ceModel =
   new CELinearModel(trainingDataForClassification);
gamma = 0.1;
pv_hat = ceModel.TrainByGD(nEpochs, gamma);
averageValidationError =
               ceModel.AverageValidationError(
                  validationDataForClassification);
            DisplayResults(nEpochs, pv_hat, averageValidationError);
        }

        public static void Dump(LabeledData[] ld)
        {
            int n_ld = ld.Length;
            for(int n = 0; n < n_ld; ++n)
            {
                Console.Write("({0:F2}, {1:F2})  ",
                    ld[n].x, ld[n].y);
                if ((n + 1) % 5 == 0) Console.WriteLine();
            }
            Console.WriteLine();
        }

        public static void DisplayResults(
            int nEpochs,
            ParameterVector2D pv,
            double averageValidationError)
        {
            Console.WriteLine("nEpochs = {0}", nEpochs);
            Console.WriteLine(
              "parameters a = {0:F2}, b = {1:F2}", pv.a, pv.b);
            Console.WriteLine(
               "Average validation error = {0:F2}",
                    averageValidationError);
        }
    }
}

// LabeledData
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Problem6
{
    struct LabeledData
    {
        public double x;  // data
        public double y;  // label or predicted value

        public LabeledData(double data, double label)
        {
            x = data;
            y = label;
        }
    }
}


// ParameterVector2D
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Problem6
{
    struct ParameterVector2D
    {
        public double a;
        public double b;
        public ParameterVector2D(
         double parameter_a, double parameter_b)
        {
            a = parameter_a;
            b = parameter_b;
        }
    }
}
```

```
// LinearModelUsingGD
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Problem6
{
    abstract class LinearModelUsingGD
    {
        abstract public double LossFunction(ParameterVector2D p);
        abstract public ParameterVector2D
            ComputeLossFunctionGradient(ParameterVector2D p);
        abstract public double Output(double x);

        protected ParameterVector2D pv;
        protected LabeledData[] td;

        public LinearModelUsingGD(LabeledData[] trainingData)
        {
            pv.a = 0.0;
            pv.b = 0.0;
            td = new LabeledData[trainingData.Length];
            for (int n = 0; n < trainingData.Length; n++)
            {
                td[n].x = trainingData[n].x;
                td[n].y = trainingData[n].y;
            }
        }

        public ParameterVector2D TrainByGD(int nEpochs, double gamma)
        {
            Random rand = new Random(168);
            ParameterVector2D p0 = new ParameterVector2D(
                (double)rand.NextDouble(), (double)rand.NextDouble());
            pv = GradientDescent(p0, nEpochs, gamma);
            return pv;
        }

        private ParameterVector2D GradientDescent(
          ParameterVector2D p0,
            int nEpochs, double gamma)
        {
            ParameterVector2D p = new ParameterVector2D(p0.a, p0.b);
            ParameterVector2D previous_p = new
              ParameterVector2D(0.0, 0.0);
            double precision = 1.0e-6;
            int epoch = 0;
            ParameterVector2D grad = new ParameterVector2D(0.0, 0.0);
            double previous_loss = 1.0e6;
            double loss = LossFunction(p);
            double loss_error = Math.Abs(loss - previous_loss);
            while (loss > precision && loss_error > precision &&
                epoch < nEpochs)
            {
                previous_p = p;
                previous_loss = loss;
                grad = ComputeLossFunctionGradient(p);
                p.a -= (grad.a * gamma);
                p.b -= (grad.b * gamma);
                loss = LossFunction(p);


                if (loss > previous_loss)
                {
                    gamma *= 0.5;
                    p = previous_p;
                    loss = previous_loss;
                }
                else
                {
                    loss_error = Math.Abs(loss - previous_loss);
                    epoch++;
                }
```

```
            }
            return p;
        }

        public double AverageValidationError(
            LabeledData[] validationData)
        {
            double error = 0.0;
            double y = 0.0;
            for (int n = 0; n < validationData.Length; ++n)
            {
                y = Output(validationData[n].x);
                error += Math.Abs(y - validationData[n].y);
            }
            error /= validationData.Length;
            return error;
        }
    }
}

// MSELinearModel
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Problem6
{
    class MSELinearModel : LinearModelUsingGD
    {
        public MSELinearModel(LabeledData[] trainingData)
            : base(trainingData) { }
        // mean square as a loss function

        // Loss Function: mean square error
        // loss = sum( (td[n].y - p.a * td[n].x - p.b)^2 ) / N
        //
```

```
public override double LossFunction(ParameterVector2D p)
{
    int n_td = td.Length;
    double loss = 0.0;
    double term = 0.0;
    for (int n = 0; n < n_td; ++n)
    {
        term = td[n].y - p.a * td[n].x - p.b;
        loss += (term * term);
    }
    loss /= n_td;
    return loss;
}


// gradient
// loss = sum( (td[n].y - p.a * td[n].x - p.b)^2 ) / N
// grad.a = 2 * sum(
//      (p.a * td[n].x + p.b - td[n].y) td[n].x  ) / N
// grad.b = 2 * sum(
//      (p.a * td[n].x + p.b - td[n].y) ) / N
//
override public ParameterVector2D ComputeLossFunctionGradient(
    ParameterVector2D p)
{
    ParameterVector2D grad = new ParameterVector2D(0.0, 0.0);
    int n_td = td.Length;
    double term = 0.0;
    for (int n = 0; n < n_td; ++n)
    {
        term = p.a * td[n].x + p.b - td[n].y;
        grad.a += (term * td[n].x);
        grad.b += term;
    }
    grad.a *= (2.0 / n_td);
    grad.b *= (2.0 / n_td);
    return grad;
}
```

```
        override public double Output(double x)
        {
            double y = pv.a * x + pv.b;
            return y;
        }
    }
}

// CELinearModel
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Problem6
{
    class CELinearModel : LinearModelUsingGD
    {
        public CELinearModel(LabeledData[] trainingData)
            : base(trainingData) { }

        // cross entropy as a loss function
        // https://en.wikipedia.org/wiki/Cross_entropy

        public static double Sigmoid(double z)
        {
            return 1.0 / (1.0 + Math.Exp(-z));
        }

        // z = p.a * td[n].x + p.b
        // y_hat = Sigmoid(z)
        // loss = -sum( td[n].y * ln y_hat +
        //              (1 - td[n].y)*ln (1 - y_hat) )
        public override double LossFunction(ParameterVector2D p)
        {
            int n_td = td.Length;
            double loss = 0.0;
```

```
   double term = 0.0;
   double z = 0.0;
   double y_hat = 0.0;
   for (int n = 0; n < n_td; ++n)
   {
      z = p.a * td[n].x + p.b;
      y_hat = Sigmoid(z);
      term = td[n].y * Math.Log(y_hat) +
         (1.0 - td[n].y) * Math.Log(1.0 - y_hat);
      loss += term;
   }
   loss /= (-n_td);
   return loss;
}

// z = p.a * td[n].x + p.b
// Sigmoid'(z) = Sigmoid(z) * (1 - Sigmoid(z))
// partial z partial p.a = td[n].x
// partial z partial p.b = 1
// y_hat = Sigmoid(z)
// y_hat_prime = Sigmoid'(z) = Sigmoid(z) * (1 - Sigmoid(z))
// partial z partial p.a = td[n].x
// partial z partial p.b = 1
// grad.a = -sum(
//   (td[n].y/y_hat + (1 - td[n].y)/(1 - y_hat))
//    y_hat_prime td[n].x  )
// grad.b = -sum(
//   (td[n].y/y_hat + (1 - td[n].y)/(1 - y_hat)) y_hat_prime  )
//
override public ParameterVector2D ComputeLossFunctionGradient(
   ParameterVector2D p)
{
   ParameterVector2D grad = new ParameterVector2D(0.0, 0.0);
   int n_td = td.Length;
   double z = 0.0;

   double y_hat = 0.0;
   double y_hat_prime = 0.0;
```

```
            double term = 0.0;
            for (int n = 0; n < n_td; ++n)
            {
                z = p.a * td[n].x + p.b;
                y_hat = Sigmoid(z);
                y_hat_prime = y_hat * (1.0 - y_hat);
                term = td[n].y / y_hat +
                        (1.0 - td[n].y) / (1.0 - y_hat);
                grad.a += (term * y_hat_prime * td[n].x);
                grad.b += (term * y_hat_prime);
            }
            grad.a *= (-1.0 / n_td);
            grad.b *= (-1.0 / n_td);
            return grad;
        }


        override public double Output(double x)
        {
            double z = pv.a * x + pv.b;
            double activation = Sigmoid(z);
            double indicator = (activation > 0.5) ? 1.0 : -1.0;
            return indicator;
        }
    }
}
```