# 通識計算機程式設計期末考

6/29/2018

試題共 7 題，兩面印製 30 頁，滿分 103

1. 當前個人電腦已經相當普及，一個人擁有兩部以上的電腦裝置，並不少見。

這些電腦裝置的作業系統未必完全相同，例如：在辦公室應用Windows，出外攜帶運行iOS的iPad。這時可能的困擾，便是如何讓iPad也能讀取Windows產生的檔案，或讓Windows讀你在iPad寫下的文稿。本題如圖1類別圖所示：主程式中，呼叫**CreateAndOpen**產生及開啟連結Windows及iOS 檔案的**TextReader陣列**，並將各個**TextReader**連結的檔案內容移入記憶緩衝器**buffer**。這是因為磁碟檔案讀寫很花時間，因此通常先將檔案內容抄到記憶緩衝器，之後的讀寫都利用記憶緩衝器，提高效率。隨後利用函式**ReadAndDisplay由buffer**讀取、顯示讀出的Windows及iOS Text檔內容。整個程式執行時的主控台畫面如圖2。當然，假設**using System;**敘述已經包含於程式中。

(a) 撰寫介面**TextReader**，宣告的函式為**FileName**、**Open**、**Close**、**Read**。**這五個函式，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%)

(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%)

(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%)

(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%)

(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();
}
```

(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%)

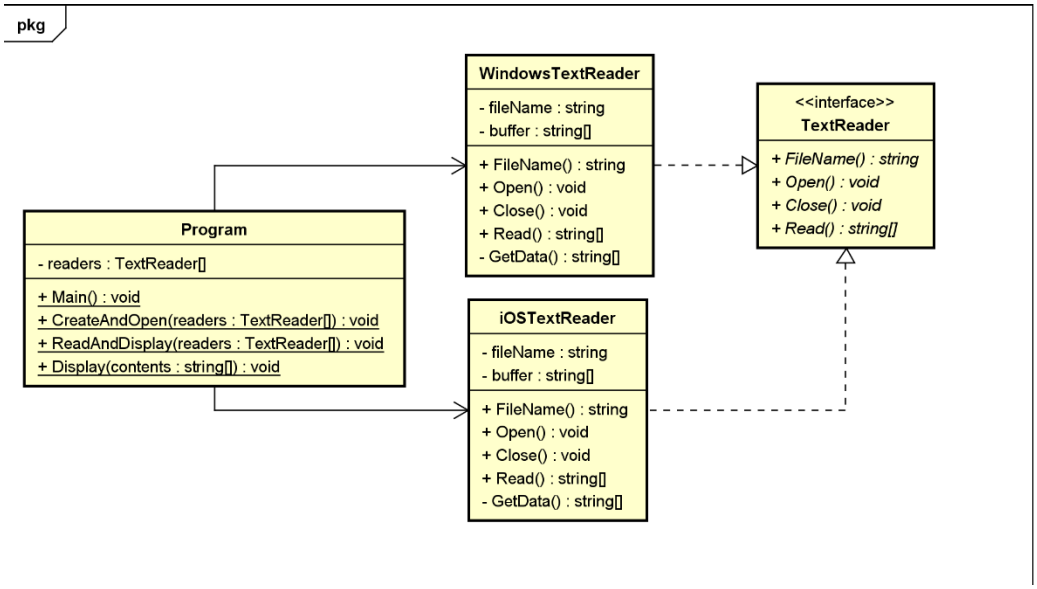


圖 1.　第 1 題對應的 UML 類別圖

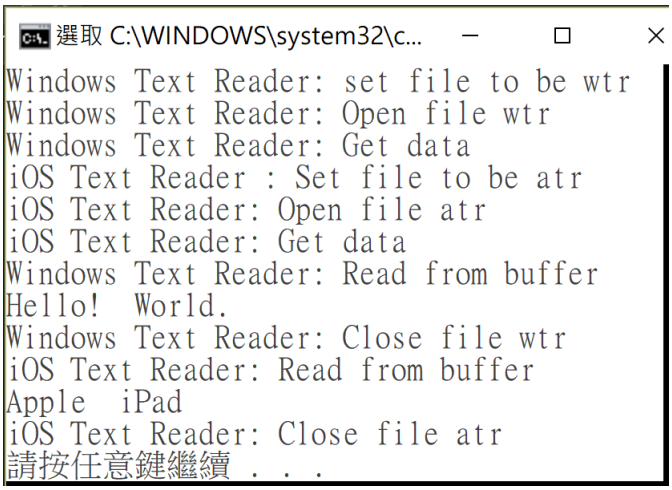


圖 2. 第 1 題程式碼，執行時的主控台螢幕畫面

2. 找出以下程式片段之錯誤，並予更正.

(a) (3%) 一個錯誤

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

(b) (3%) 一個錯誤

```
class Parents  {
    private string dad;
    private string mom;
    private int pension;
    private 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。

```
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

```
abstract class Radio  {
    abstract public void PlayMusic();
}
abstract class 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

```
class Car {
    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;
    }
    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%)

```
// Problem3
using System;
namespace Problem3  {
```

```
  class Program  {
    static void Main(string[] args)  {
      int RailLength = 6;
      int pick = 1;
      int itd;
      int coincidence;
      JeffressModel barn_owl = new JeffressModel(RailLength, pick);
      for (int t = 0; t < 10; ++t)  {
         barn_owl.FindMaxResponse(out itd, out coincidence);
         Console.WriteLine(
            "t = {0}, itd = {1}, coincidence = {2}",
             t, itd, coincidence);
          barn_owl.Update();
       }
    }
  }
}

//Jeffress Model
using System;

namespace Problem3  {
   public class JeffressModel  {
      private int railLength;
      private ForwardLine upperRail;
      private BackwardLine lowerRail;

      public JeffressModel(int railLength, int pick)  {
        this.railLength = railLength;
        int[] impulse = ImpulseSignal(pick);
        upperRail = new ForwardLine(railLength, impulse);
        lowerRail = new BackwardLine(railLength, impulse);
      }

       public void Update() {
           upperRail.Update();
           lowerRail.Update();
       }
```

```
      public int NextSample(int pickUpLoc)  {
         int outSampU = upperRail.Access(pickUpLoc);
         int outSampL = lowerRail.Access(pickUpLoc);
         int outSamp = outSampU * outSampL;
         return outSamp;
      }

       public void FindMaxRespons(out int itd, out int coincidence {
         int td = 0;
         int maxAmp = 0;
         int outSamp = 0;
         for(int pickUpLoc = 0; pickUpLoc < railLength; ++pickUpLoc){
            outSamp = NextSample(pickUpLoc);
            if(outSamp > maxAmp)  {
               maxAmp = outSamp;
               td = 2 * pickUpLoc - railLength;
             }
         }
         itd = td;
           coincidence = maxAmp;
         }

          private int[] ImpulseSignal(int pick)  {
             int[] pulse = new int[railLength];
             for (int i = 0; i < railLength; ++i)  {
                pulse[i] = 0;
             }
             pulse[pick] = 1;
             return pulse;
          }
    }
}

// DelayLine
using System;
using System.Collections.Generic;
```

```
namespace Problem3  {
  public class DelayLine  {
    protected int length;
    protected List<int> data;
    protected int pointer;

    public DelayLine(int length, int[] initialShape)  {
      this.length = length;
      data = new List<int>();
      for (int i = 0; i < length; ++i)  {
         data.Add(initialShape[i]);
      }
    }

     public virtual void Update() { }

     public int Access(int pickUpLoc)  {
        int outLoc = pointer + pickUpLoc;
        while (outLoc < 0) outLoc += length;
        while (outLoc > length - 1) outLoc -= length;
         return data[outLoc];
      }
    }

    public class ForwardLine : DelayLine  {
      public ForwardLine(int length, int[] initialShape) :
         base(length, initialShape)  {
            pointer = 0;
      }

      public override void Update()  {
        pointer = ++pointer;
        pointer = (pointer > length) ? 0 : pointer;
      }
    }

    public class BackwardLine : DelayLine  {
      public BackwardLine(int length, int[] initialShape) :
```

```
        base(length, initialShape)  {
        pointer = length - 1;
      }

      public override void Update()  {
        pointer = --pointer;
        pointer = (pointer < 0.0) ? length : pointer;
      }
    }
 }
```

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。

(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>
```

(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>
```

(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>
```

```
=====================================================================
// Problem4
using System;
using System.IO;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
namespace Problem4  {
  class Program  {
     static void Main(string[] args)  {
     try {
        string fileName = "test.aiml";
        SimpleChatbot alice = new SimpleChatbot(fileName);
        Console.WriteLine();

        BinaryFormatter formatter = new BinaryFormatter();
        FileStream output = new FileStream("alice_aiml.chat",
           FileMode.Create, FileAccess.Write);
        formatter.Serialize(output, alice);
        output.Close();
        Console.WriteLine();

        FileStream input = new FileStream("alice_aiml.chat",
        FileMode.Open, FileAccess.Read);
        Object obj = formatter.Deserialize(input);
        if (obj.GetType() == alice.GetType())  {
           SimpleChatbot bot = (SimpleChatbot)obj;
           Console.WriteLine("XML version: " + bot.XML_VERSION);
```

```
            Console.WriteLine("AIML version: " + bot.AIML_VERSION);
            Console.WriteLine("Chat category pattern: " +
               bot.PATTERNS[0]);
            Console.WriteLine("Chat category template: " +
               bot.TEMPLATES[0]);
          }
          else  {
             throw new SerializationException();
          }
        }  catch (AbnormalParsingException e)  {
          Console.WriteLine(e);
        }  catch (FileNotFoundException)  {
          Console.WriteLine("File not found");
        }  catch (SerializationException)  {
          Console.WriteLine(
           "Error in serializing/deserializing objects");
        }  catch (IOException)  {
          Console.WriteLine("Can not open or close file");
        }  catch (Exception e)  {
          Console.WriteLine(e.Message);
        }
      }
    }
}

// SimpleChatbot
using System;
using System.IO;
namespace Problem4  {
  [Serializable]
  class SimpleChatbot  {
     private string xml_version;
     private string aiml_version;
     private string[] contents;
     private string[] categories;
     private string[] patterns;
     private string[] templates;
```

```
public SimpleChatbot(string fileName)  {
try  {
  xml_version = XMLVersion(fileName);
  aiml_version = AIMLVersion(fileName);
  ParseForElements(fileName, "<aiml", "</aiml>", out contents);
  ParseForElements(contents, "<category>", "</category>",
     out categories);
  ParseForElements(categories, "<pattern>", "</pattern>",
      out patterns);
    ParseForElements(categories, "<template>", "</template>",
      out templates);
  }  catch (Exception e)  {
      Console.WriteLine(
         "Throw an exception from constructor of SimpleChatbot");
      throw e;
  }
}
 public string XML_VERSION { get { return xml_version; } }
 public string AIML_VERSION { get { return aiml_version; } }
 public string[] PATTERNS { get { return patterns; } }
 public string[] TEMPLATES { get { return templates; } }

 private string XMLVersion(string fileName)  {
    return GetVersion("<?xml", fileName, 0, 5);
 }

 private string AIMLVersion(string fileName)  {
     return GetVersion("<aiml", fileName, 1, 3);
 }

 private static string GetVersion(string tag, string fileName,
   int nSkip, int idx)  {
   string content = "";
   int length = 0;
   string line = "";
   bool beginMarkFound = false;
   bool endMarkFound = false;
   int begin = -1;
```

```
int end = -1;
char[] delimiters = new char[] { ' ', '=', '\"' };
try  {
   StreamReader input = new StreamReader(fileName);
   Console.WriteLine("GetVersion " + tag + " : Open file");
   try  {
     for(int i = 0; i < nSkip; ++i)  {
       input.ReadLine();
     }
     line = input.ReadLine();
     begin = line.IndexOf(tag);
     beginMarkFound = (begin >= 0);
     if (!beginMarkFound) {
       throw new AbnormalParsingException(
          "Symbol " + tag + " not found");
     }
     begin += tag.Length;

     end = line.IndexOf("?>");
     endMarkFound = (end >= 0);
     if (!endMarkFound)  {
        throw new AbnormalParsingException(
           "Symbol \"?>\" not found");
     }
     length = end - begin;
     if (length < 0)  {
         throw new AbnormalParsingException(
            "Abnormal" + tag + " header");
     }
     content = line.Substring(begin, length);
     string[] terms = content.Split(delimiters);
     return terms[idx];
  }
  catch (AbnormalParsingException e)  {
     Console.WriteLine(
        "Throw an abnormal-parsing exception from" +
        " GetVersion :" + tag);
     throw e;
```

```
      } catch (Exception e)  {
         Console.WriteLine(
           "Throw an exception from inner try-catch in" +
           " GetVersion : " + tag);
         throw e;
      }  finally {
         Console.WriteLine(
            "Enter inner finally in function GetVersion : " + tag);
         input.Close();
         Console.WriteLine("Close file");
      }
    }  catch (AbnormalParsingException e) {
       Console.WriteLine(
         "Throw an abnormal-parsing exception from" +
         " outer try-catch in GetVersion : " + tag);
       throw e;
    }  catch (FileNotFoundException e)  {
       Console.WriteLine(
         "Throw a file-not-found exception from" +
         " outer try-catch in GetVersion : " + tag);
       throw e;
    }  catch (Exception e)  {
       Console.WriteLine(
         "Throw an exception from" +
         " outer try-catch in GetVersion: " +tag);
       throw e;
    }
}

private void ParseForElements(
  string fileName, string beginMark, string endMark,
  out string[] elements)  {
    string content = "";
    int length = 0;
    string line = "";
    bool beginMarkFound = false;
    bool endMarkFound = false;
    int begin = -1;
```

```
int end = -1;
const int MAX_N_ELEMENTS = 10;
elements = new string[MAX_N_ELEMENTS];
int nElements = 0;

try {
   StreamReader input = new StreamReader(fileName);
   Console.WriteLine("ParseForElements: Open file");
    try {
    while (!input.EndOfStream) {
      line = input.ReadLine();
      if (beginMarkFound)  {
        begin = 0;
      }  else  {
         begin = line.IndexOf(beginMark);
         if (begin < 0) continue;
         beginMarkFound = true;
         begin = begin + beginMark.Length;
      }

      if (!endMarkFound)  {
        end = line.IndexOf(endMark);
        if(end < 0)  {
           end = line.Length - 1;
        }  else  {
           endMarkFound = true;
        }
        length = end - begin + 1;
        if (length > 0) {
          content += line.Substring(begin, length);
        }
     } else {
        beginMarkFound = false;
        endMarkFound = false;
        begin = -1;
        end = -1;
      }
    }
```

```
      elements[nElements] = content;
      nElements++;
      if (beginMarkFound && !endMarkFound)
        throw new AbnormalParsingException(
          endMark.Substring(1, endMark.Length - 1) + " not found");
    }  catch (AbnormalParsingException e) {
      Console.WriteLine(
         "Throw an abnormal-parsing exception" +
         " from ParseForElements");
      throw e;
    }  catch (Exception e) {
      Console.WriteLine(
        "Throw an exception from" +
        " inner try-catch in ParseForElements");
      throw e;
    }  finally  {
      Console.WriteLine(
        "Enter inner finally in ParseForElements");
      input.Close();
      Console.WriteLine("Close file");
     }
   }  catch (AbnormalParsingException e)  {
      Console.WriteLine(
        "Throw an abnormal-parsing exception" +
        " from outer try-catch in ParseForElements");
      throw e;
   }  catch (FileNotFoundException e)  {
      Console.WriteLine(
         "Throw a file-not-found exception" +
         " from outer try-catch in ParseForElements");
      throw e;
   }  catch (Exception e)  {
      Console.WriteLine(
        "Throw an exception from" +
        " outer try-catch in ParseForElements");
      throw e;
   }
}
```

```
private void ParseForElements(
  string[] categories, string beginMark, string endMark,
  out string[] elements) {
  string content;
  int length;
  bool beginMarkFound;
  bool endMarkFound;
  int begin;
  int end;
  const int MAX_N_ELEMENTS = 10;
  elements = new string[MAX_N_ELEMENTS];
  int nElements = 0;

  for (int i = 0; i < categories.Length; ++i)  {
  try {
    content = categories[i];
    if (content == null) continue;
    beginMarkFound = false;
    endMarkFound = false;
    begin = content.IndexOf(beginMark);
    if (begin >= 0)  {
       beginMarkFound = true;
       begin += beginMark.Length;
    }
    end = content.IndexOf(endMark) - 1;
    if (end >= 0) {
      endMarkFound = true;
      length = end - begin + 1;
      if (length > 0)  {
        elements[nElements] = content.Substring(begin, length);
        nElements++;
      }
    }
    if (beginMarkFound && !endMarkFound) {
       throw new AbnormalParsingException(
         endMark.Substring(1, endMark.Length - 1) +
         " not found");
     }
```

```
          if (!beginMarkFound && endMarkFound)  {
             throw new AbnormalParsingException(
               beginMark.Substring(1, endMark.Length - 1) +
               " not found");
          }

        }  catch(AbnormalParsingException e) {
             Console.WriteLine(
               "Enters ParseForElements with string arrays");
             Console.WriteLine(e);
        }  catch(Exception e)  {
             Console.WriteLine(
                "Enters ParseForElements with string arrays");
             Console.WriteLine(e);
        }
        }
       }
    }
}

// AbnormalParsingException
using System;
namespace Problem4  {
  public class AbnormalParsingException : ApplicationException  {
    private string message;
    public AbnormalParsingException(string message): base()  {
      this.message = message;
    }
    public override string ToString()  {
      return message;
    }
  }
}
```

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 。

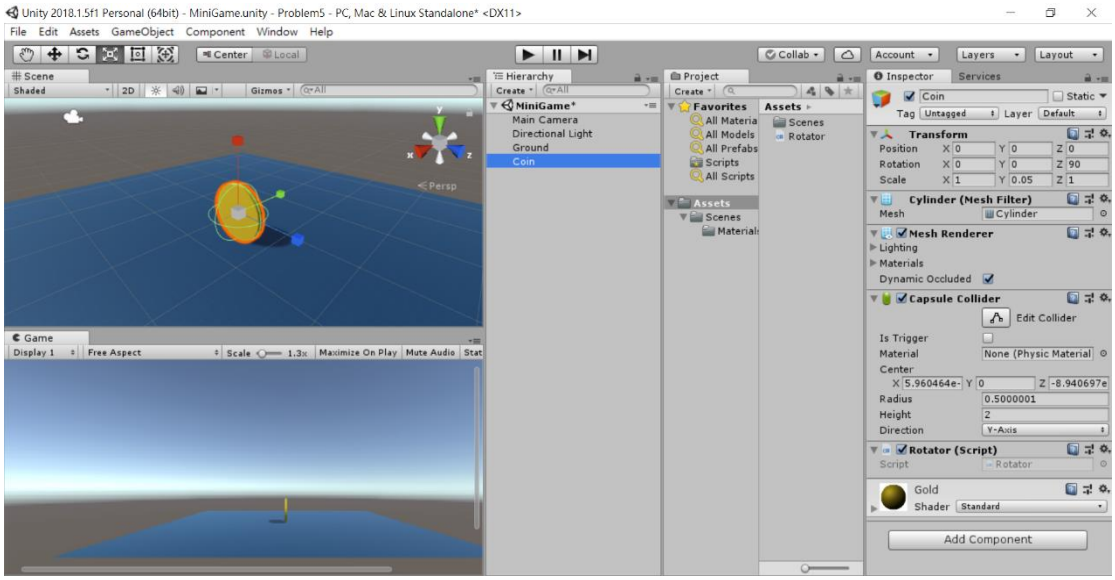

圖 3．旋轉金幣之外形與場景設計

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。

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Rotator : MonoBehaviour {
    void Start () {
       //*********************************
         (A)
       //*********************************
    }
    void Awake () {
       //*********************************
         (B)
       //*********************************
    }
    void FixUpdate () {
       //*********************************
         (C)
       //*********************************
    }
```

```
    void OnTriggerEnter(Collider other) {
        //*********************************
         (D)
        //*********************************
    }
     void Update () {
        //*********************************
         (E)
        //*********************************
    }
}
```

6. 近年來人工智慧(Artificial Intelligence, AI)的研究突飛猛進，應用日漸廣泛。深入了解後，可以知道這一波的 AI 進展，主要建構於機器學習(Machine Learning, ML)理論與技術的長足進步。簡單的說，智慧表現可以想成是一個複雜的黑箱函數。愈有「智慧」的表現，需要越多的參數，參與黑箱函數的計算。黑箱函數的參數越多，越需要更多黑箱行為的觀察數據，也就是更多輸入與對應輸出的配對資料。同時，大量的數據，也就越需要強力有效的硬體計算，才能藉由機器學習，準確求出黑箱函數中的大量參數，進而得到我們希望達成的「智慧」反應。這些需求的滿足，由於網路的發達，巨量資料可以由網際網路輕易取得，加上計算能力超強的圖形處理單元(Graphic Processing Unit, GPU)計算機架構技術成熟，已經不是太困難的問題。這些因素的湊合，導致了人工智慧領域的復興。

機器學習的概念，可以用圖 4 來說明：

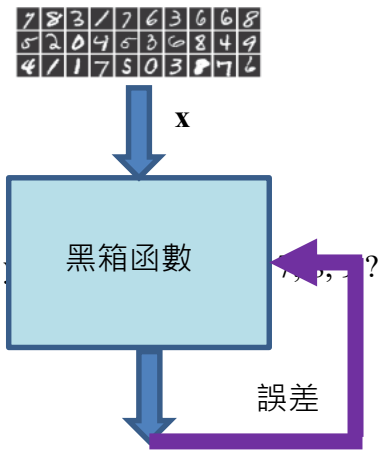


圖 4．機器學習的概念

假定要開發手寫郵遞區號的自動辨識系統，則必須先收集大量的手寫數字影像，建立資料庫(例如美國國家標準局 NIST 所建的 The MNIST of hand-writing digits, http://yann.lecun.com/exdb/mnist/)。資料庫中的每個影像，都附帶有其代表的數字數值(稱為標籤, label)。

接著設計一個以若干參數表出的黑箱函數(例如類神經網路，Artificial Neural Networks, ANN)，以手寫數字影像為輸入 $\mathbf{x}$，辨識結果為輸出 $y$。其次，將收集的影像通過黑箱函數，由黑箱函數計算出此一數字影像代表哪一個數字。由於目前黑箱函數的參數，並不能夠使黑箱函數辨識大部分手寫影像，因此對於第 $n$ 個影像資料 $\mathbf{x}_n$，其輸出 $y$ 不等於對應於 $\mathbf{x}_n$ 的應有標籤 $y_n$。

不正確的黑箱函數，藉由 $y$ 和 $y_n$ 的誤差，可以採用某種方式，調整黑箱函數內的參數內容。反覆進行如上步驟，把所有測試資料跑完一趟，稱為完成一個 epoch。進行許多個 epoch 之後，最終能使所有資料產生的誤差，大部分可以忽略。此時的黑箱函數，就具備了辨識手寫數字的「智慧」(當然，此處稱為「智能」較佳)。這種反覆修正黑箱函數參數的過程，通常稱為「訓練」(training)，或「學習」(learning)。

訓練後的黑箱函數，就不再調整函數中的參數，直接用來辨識新的手寫數字影像。這時用另外一組影像及標籤，讓黑箱函數辨識，統計其辨識成功率，就稱為驗證(validation)。驗證過程中的手寫數字影像，應該與測試資料不同，表示黑箱函數的訓練結果，具備推廣(generalization)所學知識的能力。當然，驗證的錯誤率會高於訓練資料的辨識錯誤率。猶如學生反覆練習課本習題，如果考題都出自課本，一定會得到高分；但碰到大型考試，如會考、學測，考題通常與課本習題不同，成績有很大機會，比只考兩三課習題的小考低。

訓練與驗證後的黑箱函數，隨後應用於未知標籤的實際問題，即是機器學習的「測試」(testing)階段。

實際上的機器學習，通常需要構想一個「損失函數」(loss function)，為黑箱函數參數與訓練資料的函數。進行訓練時，反覆依照損失函數的變化，調整黑箱函數參數(即是「學習」)，計算對應損失函數值。通常損失函數達到最小值時，對應的黑箱函數參數即是最能符合需求的一組參數。

從損失函數調參數的一種常用基本方法，稱為梯度下降法(gradient descent)。要說明梯度的概念，先介紹等高線：某高度的水平面，與山坡面的交線，便是山坡等高線，如圖 5 所示。

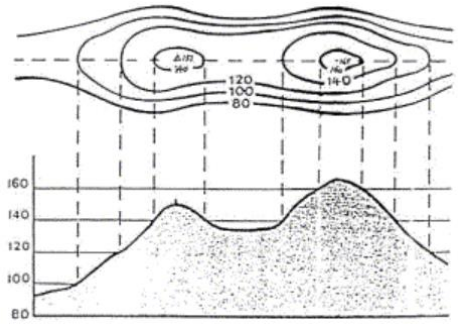


圖 5. 等高線觀念 (取自 http://anniehank.pixnet.net/blog/post/22884725-%E5%9C%8B%E4%B8%80--%E7%AD%89%E9%AB%98%E7%B7%9A%E5%9C%B0%E5%BD%A2%E5%9C%96%E5%92%8C%E5%89%96%E9%9D%A2%E5%9C%961)

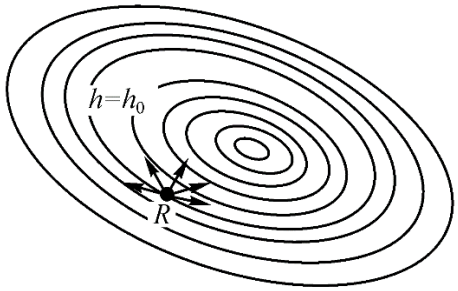


圖 6. 梯度概念

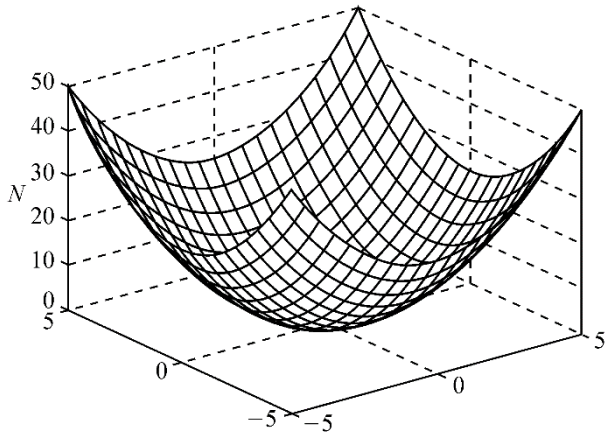


圖 7. 代表山谷的曲面

假設圖 6 代表山峰的等高線圖，且越小圈的等高線，代表越大的高度。令圖中高度$h = h_0$等高線上的一點 $R$，由 $R$ 出發，選擇不同的方向向上爬。由圖可看出：在 $R$ 點選擇垂直等高線的方向(五個箭頭標示中，中間往上的箭頭)往上爬，能夠在相同水平距離內，穿越最多條等高線，即升高最多。

所以山坡某點等高線的垂直方向向量，代表該點山高變化最陡峭的方向，稱作梯度(gradient)。

如果把圖 6 看成如圖 7 的山谷曲面等高線圖，則越小圈的等高線，代表越低的高度。這時圖 6 中，由 *R* 點下降最快的方向(垂直於等高線，面向谷底，同樣是五個箭頭標示中，中間往上的箭頭)，就是 *R* 點與梯度相反的方向。由山谷曲面上任一點出發，沿著梯度反方向前進一段距離，可以更接近谷底。反覆此一步驟，使每個步驟都沿梯度相反方向前進一小段距離，若干步後，應該就離谷底不遠。如果每個步驟都使用相同的前進距離，在接近谷底的地方，很容易會衝過頭，使高度不再穩定下降，反而升高。這時就要退回至前一步驟，並且縮短前進距離。這就是梯度下降法的概念。如果山谷曲面相當於損失函數，谷底的點座標，就是黑箱函數參數的最佳值。

以上所介紹的梯度下降法，可以寫成以下 C#程式:

```
// 假設黑箱參數只有 a 和 b
struct ParameterVector2D
{
   public double a;
   public double b;
   public ParameterVector2D(
    double parameter_a, double parameter_b)
   {
      a = parameter_a;
      b = parameter_b;
   }
}

ParameterVector2D GradientDescent(
   ParameterVector2D p0,  // 起始黑箱參數
   int nEpochs,           // 執行 epoch 數上限
   double gamma           // 每一步驟前進距離起始值
)
{
    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);

        // 主迴圈
        while (loss > precision && loss_error > precision &&
             epoch < nEpochs)
        {
           previous_p = p;
           previous_loss = loss;
           grad = ComputeLossFunctionGradient(p);   // 計算梯度
           p.a -= (grad.a * gamma);                 // 調整 a 參數
           p.b -= (grad.b * gamma);                 // 調整 b 參數
           loss = LossFunction(p);
           Console.WriteLine(
              "epoch = {0}, p = ({1}, {2}), loss = {3}  \n",
              epoch, p.a, p.b, loss);
           if (loss > previous_loss)              // 檢驗損失函數值是否上升
           {
              gamma *= 0.5;                       // 前進距離減半
              p = previous_p;                     // 回到前一點
              loss = previous_loss;
           }
           else
           {
              loss_error = Math.Abs(loss - previous_loss);
              epoch++;
           }
        }
        return p;
    }
```

其次當然就是如何定義損失函數，以及計算其梯度。一般來說，機器學習的問題，大致有兩類：迴歸(regression)與分類(classification)。上述梯度下降法程式碼，在這兩類問題都適用；但請注意，你可以修改其內容與格式，不必完全套用，以便應

用於你自己所設計的程式架構。

迴歸問題與統計學裡面的迴歸分析差不多。以圖 8 為例，假設黑箱函數為 $y = ax + b$，希望找到的參數$a, b$，能夠使直線方程式$y = ax + b$，盡量擬合(fit) $N$ 個訓練資料點 $(x_n, y_n),\ n = 1, 2, \ldots, N$ (圖中直線)。

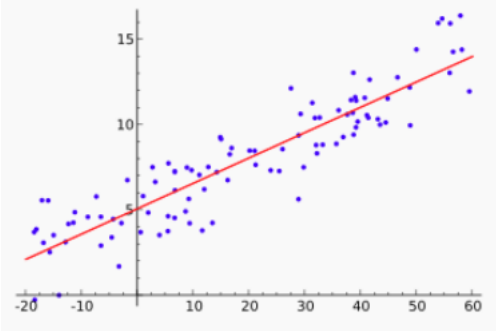


圖 8．線性回歸問題示意 (取自 https://en.wikipedia.org/wiki/Linear_regression )

這種情況下，平均平方誤差(Mean Square Error)是常用的損失函數：

$$L(a, b) = \frac{1}{N}\sum_{n=1}^{N}(y_n - ax_n - b)^2$$

如果找到的參數 $a, b$ 使損失函數最小，代表由$a, b$決定的直線 $y = ax + b$，與所有訓練點的 $y$ 座標誤差平方和最小。同時，平均平方誤差對於$a, b$的梯度(一個 **ParameterVector2D** 物件)可以導出

$$\text{grad. a} = -\frac{2}{N}\sum_{n=1}^{N} x_n(y_n - ax_n - b)$$

$$\text{grad. b} = -\frac{2}{N}\sum_{n=1}^{N}(y_n - ax_n - b)$$

以上損失函數與梯度的數學式，可以分別轉換為如下的C#程式敘述片段:

```
// loss function - mean square error
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;


    // gradient - mean square error
    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);
```

此處的 **td** 是 **LabeledData** 之 **struct** 物件陣列，而 **LabeledData** 的宣告如下：

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

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

至於分類問題，黑箱函數的目標，是把輸入的 **x** 分成兩類，其標籤 $y$ 分別以 0 和 1 表示。假定黑箱函數的參數同樣只有 $a$ 和 $b$，輸入 $x$，輸出為 $\mathrm{y} = \begin{cases} 1, & \sigma(ax+b) > 0.5 \\ 0, & \sigma(ax+b) < 0.5 \end{cases}$。此處sigmoid函數 $\sigma(z) = \frac{1}{1+e^{-z}}$，代表一個 $S$ 形狀的曲線，如圖 9 所示。由於$\sigma(ax+b)$介於 0 和 1 之間，可以解釋為標籤等於

1 的機率。其值如果大於 0.5，表示標籤為 1 的機率，高於標籤為 0 的機率 1 - $\sigma(ax+b)$。

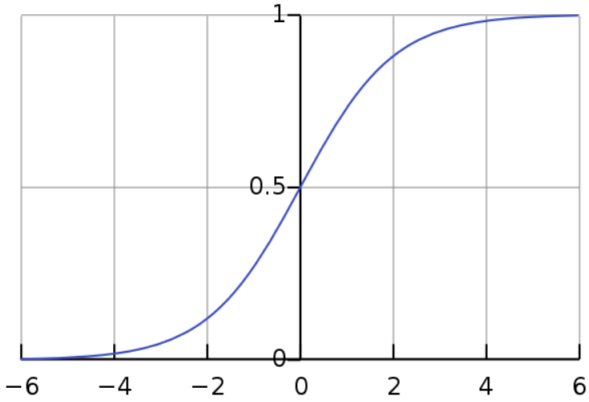


圖 9. sigmoid函數曲線(取自 https://en.wikipedia.org/wiki/Sigmoid_function)

對於這種分類問題，常用的損失函數為交叉熵(cross entropy)

$$L(a,b) = -\frac{1}{N}\sum_{n=1}^{N}[y_n \ln \hat{y}_n + (1-y_n)\ln(1-\hat{y}_n)]$$

$$\hat{y}_n = \frac{1}{1+\exp(-ax_n - b)}$$

(參看 https://en.wikipedia.org/wiki/Cross_entropy )。

對應的梯度為

$$\text{grad. a} = -\sum_{n=1}^{N}\left[\left(\frac{y_n}{\hat{y}_n} + \frac{1-y_n}{1-\hat{y}_n}\right)\hat{y}_n{}'\ x_n\right]$$

$$\text{grad. b} = -\sum_{n=1}^{N}\left[\left(\frac{y_n}{\hat{y}_n} + \frac{1-y_n}{1-\hat{y}_n}\right)\hat{y}_n{}'\ \right]$$

$$\hat{y}_n{}' = \sigma'(ax+b) = \sigma(ax+b)[1-\sigma(ax+b)]$$

轉換為 C#程式片段：

```
// loss function - cross entropy
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);


// gradient – cross entropy
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);
```

本題希望你充分應用前述內容，撰寫 C#程式：經由 10 組訓練資料，以梯度下降法，分別學習平均平方誤差與交叉熵模型，算出黑箱函數參數 $a$, $b$。再以 5 組驗證資料，求出平均驗證誤差。主控台螢幕的輸出如圖 10。兩種模型的訓練資料及驗證資料宣告如下，答案卷上可以適當註明，並直接引用，不必重新抄寫。之前提供的 **struct** 類別 **LabeledData** 與 **ParameterVector2D** 的宣告亦同。

```
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)
 };

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)
};

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)
};
```

```
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)
};
```

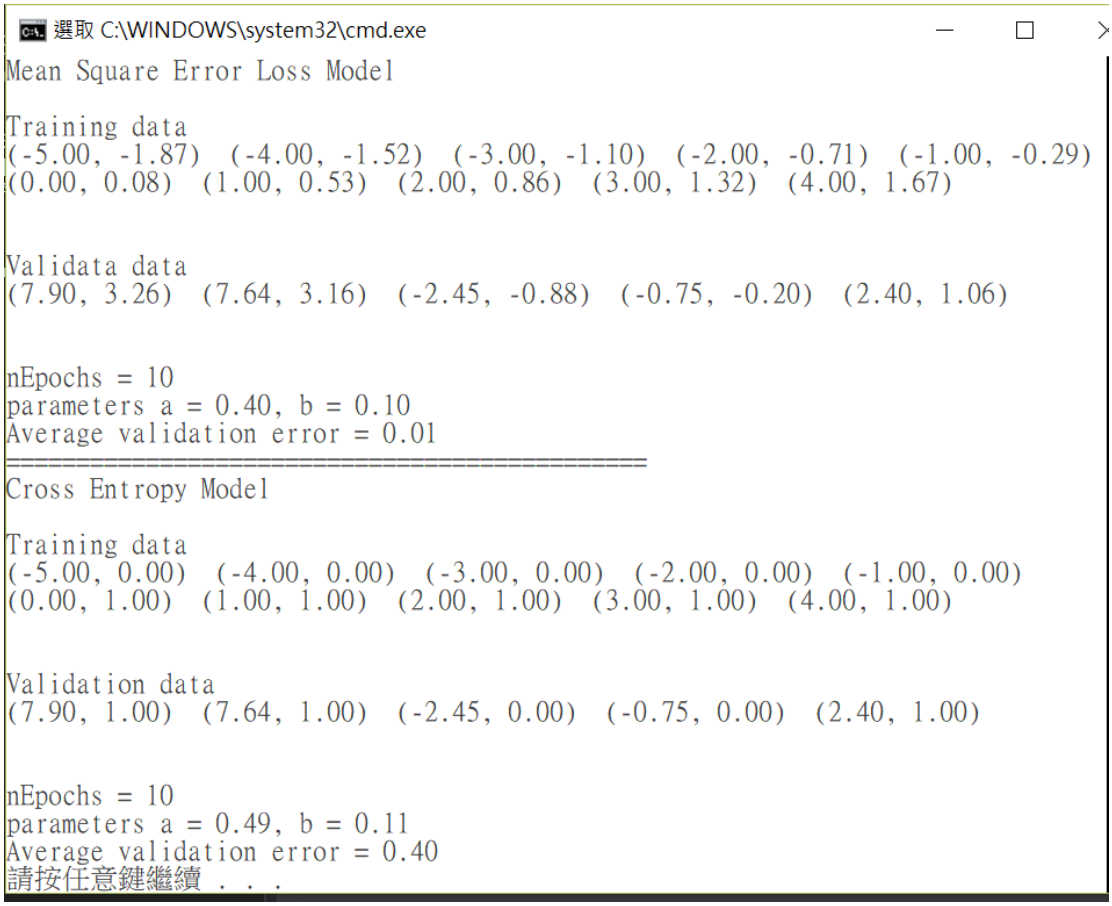


圖 10. 程式執行主控台畫面一例

本題滿分 25 分，全部程式集中寫成一個大 **Main** 函式，不區分其他函式者，最高得 18 分；善用函式者，最高得 20 分；能利用虛擬碼或 UML 類別圖思考，適當劃分類別(class)者，最高得 22 分；善用類別多型(polymorphism)者，最高得 25 分。 (25%)

# 7. 請寫下本課程教學「待改進」之處及改進方法建議。 (3%)