

# 通識計算機程式設計期中考

4/19/2019

試題共 7 題，兩面印製 16 頁，滿分 100

1. 撰寫一或數個C#敘述達成下列要求: (假設**using System;**敘述已經包含於程式中)
  - (a) 宣告 **bool** 變數 **isValid**, **int** 變數 **r**, **byte** 變數 **rByte** (3%)
  - (b) 在螢幕顯示一行字, 要求使用者輸入一個 0 至255之間的正整數 **r** (3%)
  - (c) 自鍵盤讀入一個正整數, 並將其值存入已宣告之 **int** 變數 **r** (3%)
  - (d) 將邏輯關係 **r >= 0 && r < 256** 設定給已宣告之 **bool** 變數 **isValid** (3%)
  - (e) 寫一個 **if-else** 敘述, 當已宣告之 **bool** 變數 **isValid** 為 **true** 時, 將 **r** 強制轉型為 **byte** 數值, 存入已宣告之 **byte** 變數 **rByte**. 否則在螢幕顯示 "**r**不是合格的紅色成分數值" (3%)
2. 撰寫一或數個C#敘述達成下列要求: (假設**using System;**敘述已經包含於程式中)
  - (a) 將已宣告設值之 **int** 變數 **sum**, 用算子 **+=**, 加入他處已宣告設值的 **int** 變數 **term** (3%)
  - (b) 令他處已宣告設值之 **int** 變數 **nDays**, 計算 **n** 除以 7 的商, 設定給他處已宣告之 **int** 變數 **nWeeks** (3%)
  - (c) 宣告**double**變數 **y**, 並設定其值為數學函數 $\frac{x}{\sqrt{1+x^2}}$ 。此處 **x** 用**double** 變數 **x** 代表。假定 **x** 已經於之前宣告設值 (3%)
  - (d) 利用三元運算子, 使他處已宣告設值之**double**變數 **delta**, 數值大於等於 0 時, 設定字串變數 **ds** 的值為 "實根", 反之則令 **ds** 值為 "複根"。假設 **ds** 已在他處宣告 (3%)
  - (e) 宣告變數 **ch** 為 **char** 型別, 並令其值代表倒斜線符號「\」 (3%)
3. 撰寫一或數個C#敘述達成下列要求: (假設**using System;** 敘述已經包含於程式中)
  - (a) 先以一個敘述宣告一個亂數產生器物件**rand**, 其種子數設為 777 (3%)
  - (b) 宣告三個一維 **double** 陣列 **x1**、**x2**、**w**, 其內容分別為 {0.1, 0.3, 0.5}、{0.2, 0.4, 0.6}、{1.0, 0.0, -1.0} (3%)
  - (c) 宣告整數變數 **n**, 並設初值為 **w** 的長度 (3%)
  - (d) 宣告 **double** 變數 **y**, 設其初值為 0.2。再寫一個 **for** 迴圈, 迴圈控制變數 **i** 由 0 開始, 每次加 1, 遞增至 **n-1**。迴圈每次 iteration, 以

**rand.Next()** 產生一個隨機整數；如果隨機整數是偶數，則將 **y** 值加上 **w[i]** 與 **x1[i]** 的乘積，否則將 **y** 值加上 **w[i]** 與 **x2[i]** 的乘積。迴圈結束後，以 **y** 為參數，呼叫 **static double** 函式 **ElliotSig**，將函式值設為 **double** 變數 **output** (3%)

(e) 寫一個 **static double** 函式 **ElliotSig**，設其輸入參數為 **double y**，傳回算式  $\frac{y}{1+|y|}$  之值。注意絕對值 **|y|** 可以呼叫 **Math.Abs(y)** 來計算 (3%)

4. 找出以下程式片段之錯誤，並在盡量保持原先程式碼之前提下，予以更正。  
 假設 **using System;** 敘述已經包含於程式中。

(a) (3%) (一個語法錯誤) 執行時螢幕應顯示



```
int a = 1;
int b = 1;
int c = 1;
int[,] I = { {a, 0, 0},
              {0, b, 0},
              {0, 0, c}};


if( a == b == c )
{
    Console.WriteLine("I is an identity matrix");
}
else
{
    Console.WriteLine(
        "Diagonal matrix I is not an identity matrix");
}
```

(b) (3%) (一個語義錯誤) 執行時螢幕應顯示



```
int[] x = {0, 1, 2, 3};
int i = 0;
while(i <= 4)
{
    Console.WriteLine("x[" + i + "]=" + x[i]);
    ++i;
}
```

(c) (3%) (一個語義錯誤) 執行時螢幕應顯示

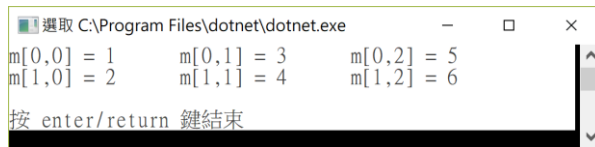


```

double p0 = 10000.0;
double loss_rate = 0.1;
double p = p0;
int nYears;
for(nYears = 1; nYears < 4; ++nYears)
{
    p = p*(1.0 - loss_rate);
    if(p < 0.5*p0) break;
}
Console.WriteLine(
    "The principal is halved after {0} years", nYears);

```

(d) (3%) (一個語義錯誤) 執行時螢幕應顯示:



```

選取 C:\Program Files\dotnet\dotnet.exe
m[0,0] = 1      m[0,1] = 3      m[0,2] = 5
m[1,0] = 2      m[1,1] = 4      m[1,2] = 6
按 enter/return 鍵結束

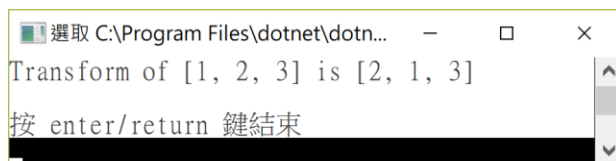
```

```

int[,] m = { {1, 3, 5}, {2, 4, 6}};
for(int i = 0; i < 3; ++i)
{
    for(int j = 0; j < 2; ++j)
    {
        Console.Write(
            "m[" + i + "," + j + "] = " + m[i, j] + "\t");
    }
    Console.WriteLine();
}

```

(e) (3%) (一個語義錯誤) 執行時螢幕應顯示:



```

選取 C:\Program Files\dotnet\dotn...
Transform of [1, 2, 3] is [2, 1, 3]
按 enter/return 鍵結束

```

```

using System;
using System.Diagnostics;
. . . . .
static void Main(string[] args)
{
    int[] vector_a = {1, 2, 3};
    int[] vector_c = Transform(ref vector_a);
    Debug.Assert(vector_a.Length == 3);
    Console.WriteLine(
        "Transform of [{0}, {1}, {2}] is [{3}, {4}, {5}]",

```

```

        vector_a[0], vector_a[1], vector_a[2],
        vector_c[0], vector_c[1], vector_c[2]);
    }
    static int[] Transform(ref int[] vector_a)
    {
        vector_a = new int[] {
            vector_a[0], vector_a[1], vector_a[2], 1};
        int[,] t = { {0, 1, 0, -1},
                     {1, 0, 0, 1},
                     {0, 0, 1, 0},
                     {0, 0, 0, 1}};
        int[] vector_c = {0, 0, 0, 1};
        for(int i = 0; i < 3; ++i)
        {
            vector_c[i] = 0;
            for(int j = 0; j < 3; ++j)
            {
                vector_c[i] += t[i, j]*vector_a[j];
            }
        }
        double den = (double) (vector_c[3]);
        int[] result = new int[] {
            (int) (vector_c[0]/den),
            (int) (vector_c[1]/den),
            (int) (vector_c[2]/den)};
        return result;
    }
}

```

## 5. 試寫出下列程式的螢幕輸出 (5 %)

```

using System;

namespace Problem5
{
    class Program
    {
        static void Main(string[] args)
        {
            double x = 0.2;
            double p_prev2 = 1.0;
            double p_prev1 = x;
            Console.WriteLine("P( 0, " + x + " ) = " + p_prev2);
            Console.WriteLine("P( 1, " + x + " ) = " + p_prev1);

            double p = 0.0;

```

```

        for(int nu = 2; nu < 5; ++nu)
        {
            p = (2.0 * nu + 1)*x*p_prev1 - nu * p_prev2;
            p /= (nu + 1);
            p_prev2 = p_prev1;
            p_prev1 = p;

            Console.WriteLine("P( " + nu + ", " + x + " ) = " + p);
        }
        // ending the program
        Console.WriteLine();
        Console.WriteLine("按 enter/return 鍵結束");
        Console.ReadLine();
    }
}

```

## 6. 試寫出下列程式的螢幕輸出 (10 %)

(本題內容修改自James McCaffrey的部落格文章：測試回合 - 了解 LSTM 儲存格 (使用 C#)

<https://msdn.microsoft.com/zh-tw/magazine/mt846470.aspx>)

```

using System;
namespace Problem6
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("\nBegin Simplified LSTM cell IO demo \n");

            Console.WriteLine(
                "Creating an n=2 input, m=3 state LSTM cell");
            Console.WriteLine("Sending input = (1.0, 2.0) to LSTM \n");

            double[,] xt = { {1.0},
                              {2.0} };
            double[,] h_prev = { {0.0},
                                  {0.0},
                                  {0.0} };
            double[,] c_prev = { {0.0},
                                  {0.0},
                                  {0.0} };
            double[,] W = { {-0.1, 0.2},
                             {0.5, -0.4},
                             {0.5, -0.1} };
            double[,] U = { {0.2, -0.8, 0.4},
                             {-0.1, 0.2, 0.1},
                             {-0.2, 0.4, -0.1} };
            double[,] b = { {0.1},
                             {0.2},
                             {-0.1} };

            double[,] Wf = MatrixCopy(W);
            double[,] Wi = MatrixCopy(W);
            double[,] Wo = MatrixCopy(W);
            double[,] Wc = MatrixCopy(W);

            double[,] Uf = MatrixCopy(U);
            double[,] Ui = MatrixCopy(U);

```

```

double[,] Uo = MatrixCopy(U);
double[,] Uc = MatrixCopy(U);

double[,] bf = MatrixCopy(b);
double[,] bi = MatrixCopy(b);
double[,] bo = MatrixCopy(b);
double[,] bc = MatrixCopy(b);

double[,] ht;
double[,] ct;

ComputeOutputs(xt, h_prev, c_prev,
    Wf, Wi, Wo, Wc, Uf, Ui, Uo, Uc, bf, bi, bo, bc,
    out ht, out ct);

Console.WriteLine("Output is:");
MatrixDump(ht);
Console.WriteLine("New cell state is:");
MatrixDump(ct);
Console.WriteLine("End LSTM demo ");

// ending the program
Console.WriteLine();
Console.WriteLine("按 enter/return 鍵結束");
Console.ReadLine();
}

static void ComputeOutputs(
    double[,] xt, double[,] h_prev, double[,] c_prev,
    double[,] Wf, double[,] Wi, double[,] Wo, double[,] Wc,
    double[,] Uf, double[,] Ui, double[,] Uo, double[,] Uc,
    double[,] bf, double[,] bi, double[,] bo, double[,] bc,
    out double[,] ht, out double[,] ct
)
{
    double[,] s = MatrixSum(
        MatrixProd(Wf, xt), MatrixProd(Uf, h_prev), bf);
    double[,] ft = MatrixReLU(s);
    s = MatrixSum(MatrixProd(Wi, xt), MatrixProd(Ui, h_prev), bi);
    double[,] it = MatrixReLU(s);
    s = MatrixSum(MatrixProd(Wo, xt), MatrixProd(Uo, h_prev), bo);
    double[,] ot = MatrixReLU(s);
    s = MatrixSum(MatrixProd(Wc, xt), MatrixProd(Uc, h_prev), bc);
    double[,] sh1 = MatrixHada(ft, c_prev);
    double[,] sh2 = MatrixHada(it, MatrixReLU(s));
    ct = MatrixSum(sh1, sh2);
    ht = MatrixHada(ot, MatrixReLU(ct));
}

// Matrix routines
static double[,] MatrixCreate(int nRows, int nCols)
{
    double[,] result = new double[nRows, nCols];
    for(int i = 0; i < nRows; ++i)
    {
        for(int j = 0; j < nCols; ++j)
        {
            result[i, j] = 0.0;
        }
    }
    return result;
}

```

```

}

static double[,] MatrixCopy(double[,] m)
{
    int nRows = m.GetUpperBound(0)+1;
    int nCols = m.GetUpperBound(1)+1;
    double[,] result = new double[nRows, nCols];
    for(int i = 0; i < nRows; ++i)
    {
        for(int j = 0; j < nCols; ++j)
        {
            result[i, j] = m[i, j];
        }
    }
    return result;
}

static double[,] MatrixProd(double[,] a, double[,] b)
{
    int nRows_a = a.GetUpperBound(0)+1;
    int nCols_a = a.GetUpperBound(1)+1;
    int nRows_b = b.GetUpperBound(0)+1;
    int nCols_b = b.GetUpperBound(1)+1;
    if(nCols_a != nRows_b)
        throw new Exception(
            "imcompatibe dimensions of matrices for MatrixProd");

    double[,] result = MatrixCreate(nRows_a, nCols_b);
    for(int i = 0; i < nRows_a; ++i)
    {
        for(int j = 0; j < nCols_b; ++j)
        {
            result[i, j] = 0.0;
            for(int k = 0; k < nCols_a; ++k)
            {
                result[i, j] += a[i, k]*b[k, j];
            }
        }
    }
    return result;
}

// element-wise functions
static double ReLu(double x)
{
    double result = (x >= 0)? x : 0;
    return result;
}

static double[,] MatrixReLU(double[,] m)
{
    int nRows = m.GetUpperBound(0)+1;
    int nCols = m.GetUpperBound(1)+1;
    double[,] result = MatrixCreate(nRows, nCols);
    for(int i = 0; i < nRows; ++i)
    {
        for(int j = 0; j < nCols; ++j)
        {
            result[i, j] = ReLu(m[i, j]);
        }
    }
}

```

```

        return result;
    }

    // Hadamard element-wise multiplication
    static double[,] MatrixHada(double[,] a, double[,] b)
    {
        int nRows_a = a.GetUpperBound(0)+1;
        int nCols_a = a.GetUpperBound(1)+1;
        int nRows_b = b.GetUpperBound(0)+1;
        int nCols_b = b.GetUpperBound(1)+1;
        if( nRows_a != nRows_b || nCols_a != nCols_b )
            throw new Exception(
                "imcompatibe dimensions of matrices for MatrixHada");

        double[,] result = MatrixCreate(nRows_a, nCols_a);
        for(int i = 0; i < nRows_a; ++i)
        {
            for(int j = 0; j < nCols_a; ++j)
            {
                result[i, j] = a[i, j]*b[i, j];
            }
        }
        return result;
    }

    static double[,] MatrixSum(double[,] a, double[,] b)
    {
        int nRows_a = a.GetUpperBound(0)+1;
        int nCols_a = a.GetUpperBound(1)+1;
        int nRows_b = b.GetUpperBound(0)+1;
        int nCols_b = b.GetUpperBound(1)+1;
        if( nRows_a != nRows_b || nCols_a != nCols_b )
            throw new Exception(
                "imcompatibe dimensions of matrices for MatrixSum");

        double[,] result = MatrixCreate(nRows_a, nCols_a);
        for(int i = 0; i < nRows_a; ++i)
        {
            for(int j = 0; j < nCols_a; ++j)
            {
                result[i, j] = a[i, j] + b[i, j];
            }
        }
        return result;
    }

    static double[,] MatrixSum(double[,] a, double[,] b, double[,] c)
    {
        int nRows_a = a.GetUpperBound(0)+1;
        int nCols_a = a.GetUpperBound(1)+1;
        int nRows_b = b.GetUpperBound(0)+1;
        int nCols_b = b.GetUpperBound(1)+1;
        int nRows_c = c.GetUpperBound(0)+1;
        int nCols_c = c.GetUpperBound(1)+1;
        if( nRows_a != nRows_b || nCols_a != nCols_b ||
            nRows_b != nRows_c || nCols_b != nCols_c )
            throw new Exception(
                "imcompatibe dimensions of matrices for MatrixSum");

        double[,] result = MatrixCreate(nRows_a, nCols_a);
        for(int i = 0; i < nRows_a; ++i)

```



```

        {
            for(int j = 0; j < nCols_a; ++j)
            {
                result[i, j] = a[i, j] + b[i, j] + c[i, j];
            }
        }
        return result;
    }

    static void MatrixDump(double[,] m)
    {
        int nRows = m.GetUpperBound(0)+1;
        int nCols = m.GetUpperBound(1)+1;
        for(int i = 0; i < nRows; ++i)
        {
            for(int j = 0; j < nCols; ++j)
            {
                Console.Write("{0:F4} \t", m[i, j]);
            }
            Console.WriteLine();
        }
        Console.WriteLine();
    }
}
}
}

```

7. 本題的背景說明修改自(陳彥谷, *失智老人陪伴機器人之華語對話系統*, 國立台灣大學電機工程學研究所, 碩士論文, 2018.)。該論文並提供更多相關研究的簡介, 可進一步參考。

對當下的人工智慧浪潮, 如何認定電腦已經具有人工智能, 是一個有趣而且重要的課題。早在 1950 年代, 計算機理論之父 Turing 便已提出著名的 Turing test: 令電腦程式與真人分在不同封閉房間, 透過電傳打字機, 與人類裁判對話。如果裁判在一定夠長時間內, 無法判斷談話對象是程式還是真人, 則可斷定電腦程式具有智能(A. M.Turing, “Computing machinery and intelligence,” *Mind*, vol. 59, pp. 433–460, 1950.)。

受此觀念啟發, Joseph Weizenbaum 於 1966 年發表模仿心理治療師, 與病人進行所謂「心理治療」的對話機器人 ELIZA (J.Weizenbaum, “ELIZA—a computer program for the study of natural language communication between man and machine.,” *Commun. ACM*, vol. 9, pp. 36–45, 1966.)。ELIZA 會依據設定的關鍵字, 產生不同的回應, 也常會使用相當空泛的語句, 如真人心理治療師所為, 維持對話的進行。這樣的簡單作法過程, 卻令人驚奇發現, 相當容易製造機器能夠對話的假象, 甚至使 Weizenbaum 的秘書會和 ELIZA 談私密的心事。當然, ELIZA 未能通過 Turing test, 但卻是當時, 最具類似人對話能力的對話機器人。時至今日, 依舊有以此為基礎的聊天機器人(chatbot)研發專案。

為測試並加速聊天機器人的進展，1990 年開始的 Loebner Prize 比賽<sup>12</sup>，每年舉辦一次，選出當年最像人的對話機器人。比賽採用 Turing test，雖然有過一些爭議，但還是目前最具代表性的對話機器人競賽。2011 年終於出現第一隻通過 Turing test 的聊天機器人 Cleverbot：測試中，它成功地使近六成的人類評審，誤以為他是真人。Cleverbot 特別之處，在於它被放到網路上，收集大量的對話與回應，並從中分析找出關鍵字。當 Cleverbot 被問到相似問題時，就從儲存的對話庫中選取回應。之前及其他 chatbot，多半採用由 ELIZA 衍生的方法：開發人員為 chatbot 設計對話，並且運用規則及 template，決定 chatbot 的對話語句。

隨著機器學習(machine learning, ML)的蓬勃發展，已有許多研究將之用於聊天機器人及智慧音箱的開發：包括 Apple Siri<sup>3</sup>、api.ai<sup>4</sup>、Facebook 的 Wit.ai<sup>5</sup>、Amazon Alexa<sup>6</sup> 以及 Microsoft LUIS<sup>7</sup>。網路上還流傳 Google assistant，宛若真人，向當地髮廊與餐廳訂位的過程影片<sup>8</sup>。

中文華語對話機器人的研究較少，而且中文與英文的文法，以及用語差異相當大。英語系國家可行的對話系統，很難直接套用至華語。台大電資學院與中央研究院有多年相關於中文語音的研究，知名科技廠商如華碩，甚至推出了有初步華語對話功能的居家機器人 Zenbo<sup>9</sup>。而中國的科研人員及網路軟體業者也有許多研究成果，包括智慧喇叭「天貓精靈 X1<sup>10</sup>」等。

雖然機器學習是如今對話機器人研發的主流，類似 ELIZA，以對話規則和 template 建構的簡單原型聊天機器人，仍有其教學價值。所以網路上可以找到一些不同版本的 ELIZA 實作解說，例如金門大學陳鍾誠教授的部落格文章<sup>11</sup>，Charles Hayden 的 Eliza Test<sup>12</sup>等。

本題以開發類似簡易 ELIZA 的對話程式為目標，程式名稱為 Seliza (Simplified eliza)，其架構參考陳鍾誠教授的 Java 實作<sup>13</sup>。我們將 Seliza 的開發，規劃為以下三個版本：

V0.1: 擷取對話者語句中的關鍵字，從回應語句庫直接選取對應回答。例如，對方說：「謝謝」，Seliza 先由關鍵字表，查出其為第 0 個關鍵字，

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Loebner\\_Prize](https://en.wikipedia.org/wiki/Loebner_Prize)

<sup>2</sup> <https://medium.com/pandorabots-blog/mitsuku-wins-loebner-prize-2018-3e8d98c5f2a7>

<sup>3</sup> <https://en.wikipedia.org/wiki/Siri>

<sup>4</sup> <https://api.ai/>

<sup>5</sup> <https://wit.ai/>

<sup>6</sup> <https://developer.amazon.com/alexa>

<sup>7</sup> <https://www.luis.ai/home>

<sup>8</sup> <https://www.youtube.com/watch?v=D5VN56jQMWM>

<sup>9</sup> <https://zenbo.asus.com/tw/>

<sup>10</sup> <https://bot.tmall.com/>

<sup>11</sup> <http://ccckmit.wikidot.com/code:eliza>

<sup>12</sup> <http://www.chayden.net/eliza/Eliza.html>

<sup>13</sup> <http://ccckmit.wikidot.com/code:eliza>

便由語句庫找出第0列三個可能回應，隨機選取其中之一的「不客氣」來回應。如果對方輸入字串中，都找不到關鍵字表所列的關鍵字，就從一個語義空泛的語句表中，隨機選取一個句子回答。

V0.2: 除了V0.1功能外，另有一個語句庫，其回答包括 template，要由對話者輸入語句中，找出對應的部分語句，填入 template 中。例如，對方說：「我覺得計程的考試很難」，Seliza 的回應：「為什麼你覺得計程的考試很難?」。這樣的對話模式可以寫成：「我覺得\*」，「為什麼你覺得\*」；其中的「\*」符號，就是一個template，可以用不同的內容替代：例如，「我覺得很沮喪」，「為什麼你覺得很沮喪?」，其 template 便是「很沮喪」。利用template，可以處理更大範圍的對話。

V0.3: 除了V0.2功能外，要能在對話者的template語句包括「你」或「我」時，在回應語句中將其轉換為「我」或「你」。例如：「我認為我的成績比你」，「為什麼你的成績比我好」。

假設上述三個版本用到的關鍵字及語句庫，分別已建立為一維陣列和二維矩形陣列，宣告於主程式開頭的部分。同時假定某一關鍵字，對應的回覆語句有三個，但可能重複。如以下程式片段所示：

```
namespace Problem7
{
    public class Program
    {
        static void Main(string[] args)
        {
            string[] key_words = {
                "謝謝", "對不起", "抱歉", "不好意思", "我是", "甚麼", "什麼",
                "何時", "誰", "哪裡", "如何", "為何", "原因", "理由",
                "你好", "嗨", "或許", "不曉得", "不知道", "總是", "常常",
                "像", "對", "朋友", "電腦", "難過", "高興"
            };

            string[,] responses = {
                {"不客氣!", "不客氣!", "不客氣!"}, {"沒問題", "不會", "沒問題",
                {"沒問題", "不會", "沒問題"}, {"沒問題", "不會", "沒問題"},
                {"你好, 久仰, 久仰", "你好, 我是Seliza", "你好"},
                {"你的答案是甚麼?", "為甚麼你對這問題有興趣?",
                "何不問問別人?"},
                {"你的答案是甚麼?", "為甚麼你對這問題有興趣?",
                "何不問問別人?"},
            };
        }
    }
}
```

```

{"你認為是甚麼時間?", "為甚麼你對這問題有興趣?",
 "何不問問別人?"},
{"你認為是誰?", "為甚麼你對這問題有興趣?", "何不問問別人?"},
{"你認為是哪裡?", "為甚麼你對這問題有興趣?", "何不問問別人?"},
{"你的答案是甚麼?", "為甚麼你對這問題有興趣?",
 "何不問問別人?"},
{"你認為是甚麼原因?", "為甚麼你對這問題有興趣?",
 "何不問問別人?"},
{"真的?", "有其他原因嗎?", "真的?"},
{"這說明了甚麼?", "有其他理由嗎?", "真的?"},
{"你好, 近來可好?", "你好, 高興看到你", "你好."},
{"嗨, 近來可好?", "嗨, 高興看到你", "你好."},
{"你好像不太確定?", "或許吧.", "或許吧."},
{"訊息不夠嗎?", "可能要問專家吧", "資訊不夠嗎?"},
{"資訊不夠嗎?", "可能可以問問內行的人吧", "訊息不夠嗎?"},
{"可以說得具體一點嗎?", "甚麼時候?", "甚麼時候?"},
{"舉個例子.", "說說看, 甚麼時候?", "甚麼時候?"},
{"有多像?", "哪裡最像?", "是嗎?"},
{"確定?", "了解", "噢!"},
{"多告訴我一些他的事", "你們認識多久?", "交情好嗎?"},
{"你說的電腦是我嗎?", "你說的電腦是我嗎?",
 "你說的電腦是我嗎?"},
{"別想那麼多", "別難過, 事情總會解決",
 "有甚麼我可以幫忙的嗎?"},
{"不錯", "好極了", "真棒"}
};

string[] responses_without_matched_keywords = {
    "我了解", "還有問題嗎 ?", "請繼續說", "可以說的更詳細嗎?",
    "這樣喔! 我知道了!", "然後呢? 發生甚麼事?",
    "再來呢? 可以多說一些嗎?", "接下來呢? ",
    "然後呢?", "多談談你自己, 好嗎?", "祝福你"
};

string[] key_words_for_templated_responses = {
    "可不可以", "我想", "你是", "覺得", "認為", "以為", "不能",
    "不想", "不希望", "不", "請", "你"
};

string[,] responses_with_template = {
    {"你確定要", "你確定要", "你確定要"},

```

```

        {"你確定想", "你確定想", "你確定想"},
        {"你認為我是", "你認為我是", "你認為我是"},
        {"為什麼你覺得", "是嗎? 為什麼", "為什麼" },
        {"為什麼你認為", "是嗎? 為什麼", "為什麼" },
        {"為什麼你以為", "是嗎? 為什麼", "為什麼" },
        {"為什麼不能", "試一下, 或許就能", "再試一下"},
        {"為什麼不想", "試一下, 或許就能", "再試一下"},
        {"為什麼不希望", "試一下, 或許就能", "再試一下"},
        {"為什麼不", "為什麼不", "為什麼不"},
        {"我該怎樣", "是否想要我", "為什麼要我"},
        {"為甚麼這麼關心我", "你自己", "為甚麼這麼關心我"}
    };
    //*****
[+] 請填入適當程式碼
    //*****
}
}
}

```

其中 `key_words`、`responses`、`responses_without_matched_keywords` 對應V0.1中的關鍵字表、回應語句庫、沒有關鍵字符合時的回應語句庫。而 `key_words_for_templated_responses`、`responses_with_template` 則是V0.2中考慮 `template` 效果的關鍵字及回應語句庫。注意此處的關鍵字如“可不可以”代表「可不可以\*」，對應的回應“你確定要”，就是「你確定要\*」，都省略了代表 `template` 的「\*」。同時注意，此處的 `template` 對應到使用者輸入語句關鍵字之後的部分，所以也稱為 `tail`。

V0.1至V0.3的程式輸出畫面實例如圖 1 至圖 3。

```
選擇 C:\Program Files\dotnet\dotnet.exe
Selina: Hi, 請問如何稱呼您?
鄭士康
Selina: 您好, 請逐行按鍵輸入對話, 要結束就說"再見"
鄭士康: 抱歉, 打攪你.
Selina: 不會
鄭士康: 為什麼電腦會懂得我在說甚麼?
Selina: 何不問問別人?
鄭士康: 要問誰?
Selina: 你認為是誰?
鄭士康: 或許是工程師吧.
Selina: 或許吧.
鄭士康: 電腦越來越厲害了.
Selina: 你說的電腦是我嗎?
鄭士康: 是啊.
Selina: 接下來呢?
鄭士康: 沒事了.
Selina: 請繼續說
鄭士康: 不用了.
Selina: 可以說的更詳細一點嗎?
鄭士康: 以後再說吧.
Selina: 祝福你
鄭士康: 謝謝.
Selina: 不客氣!
鄭士康: 再見.
Selina: 這樣喔! 我知道!
鄭士康: 再見
Selina: 再見!

按 enter/return 鍵結束
```

圖1. Seliza V0.1 的對話實例 (圖中的Selina是Seliza之誤)

```
選擇 C:\Program Files\dotnet\dotnet.exe
Seliza: Hi, 請問如何稱呼您?
鄭士康
Seliza: 您好, 請逐行按鍵輸入對話, 要結束就說"再見"
鄭士康: 我覺得電腦總有一天會比人類聰明
Seliza: 為什麼你覺得電腦總有一天會比人類聰明
鄭士康: 理由是人類科技進步的越來越快
Seliza: 有其他理由嗎?
鄭士康: 沒有了
Seliza: 多談談你自己, 好嗎?
鄭士康: 算了
Seliza: 這樣喔! 我知道!
鄭士康: 你知道甚麼?
Seliza: 為甚麼這麼關心我知道甚麼?
鄭士康: 想知道電腦有多厲害
Seliza: 你說的電腦是我嗎?
鄭士康: 是啊
Seliza: 這樣喔! 我知道!
鄭士康: 好吧, 下次再說
Seliza: 還有問題嗎?
鄭士康: 沒了
Seliza: 多談談你自己, 好嗎?
鄭士康: 再見
Seliza: 再見!

按 enter/return 鍵結束
```

圖2. Seliza V0.2 對話實例

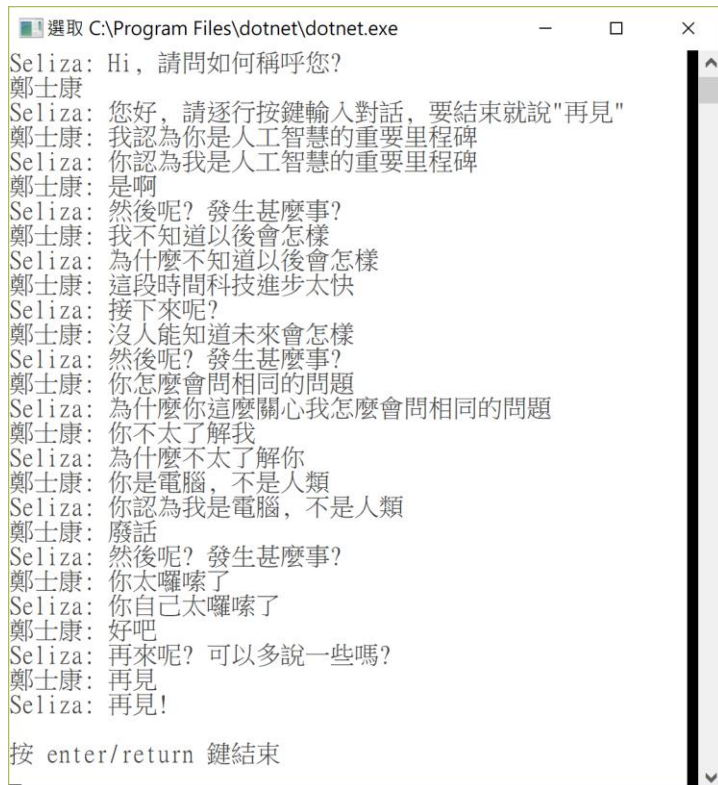


圖3. Seliza V0.3 對話實例

本題滿分 25 分，全部程式集中寫成一個大 **Main** 函式，不區分函式者，最高得 20 分；善用函式，乃至尚未教到的物件導向程式設計(object-oriented programming)者，最高得 23 分；能利用虛擬碼或流程圖思考，適當劃分函式或類別(class)者，最高得 25 分(使用虛擬碼)或 24 分(使用流程圖)。未完成最後版本者，成績酌打折扣。(25%)

陣列 `key_words`、`responses`、`responses_without_matched_keywords`、`key_words_for_templated_responses`、`responses_with_template` 都可以直接引用，不必宣告。此外，以下的 C# 程式敘述可以參考利用：

```
string input;
. . . . .
// 檢查字串 input 是否包含字串"再見"
int idx = input.IndexOf("再見");
// idx < 0 時，表示input不包含字串"再見"
// idx >= 0 時，idx是字串"再見"在input中的開始位置
. . . . .
// 取出字串input中，由位置 k 到最後的子字串
string tail = input.Substring(k);
. . . . .
```

```

// 將字串input分解為字元，順序放入字元陣列 buffer 之中
char[] buffer = input.ToCharArray();
. . . . .
// 由字元陣列 buffer 建構字串 answer
string answer = new string(buffer);
. . . . .
// 合併字串 answer 及 tail，成為新的字串 answer
answer = answer + tail;
. . . . .
// 由 startIdx 位置開始，搜尋字元'我'在字串input出現的位置
idx = input.IndexOf('我', startIdx);
// idx < 0 時，表示input中，startIdx之後，不包含字元'我'
// idx >= 0 時，idx是字元'我'在input中，startIdx之後的位置
. . . . .
//在回應之前，先暫停 2 秒(2000毫秒)，模仿按鍵打字所需時間
//避免反應太快，容易被看出是電腦的回應
Thread.Sleep(2000); // to simulate time for human typing
Console.WriteLine("Seliza: " + answer);
//注意使用Thread.Sleep，需要在程式碼前，加上 using System.Threading; 敘述

```