

通識計算機程式設計期中考

4/27/2018

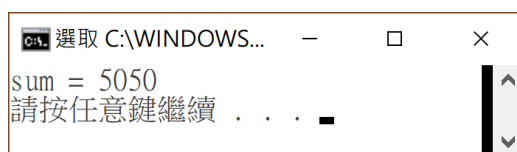
試題共 7 題，兩面印製 14 頁，滿分 100

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 **isEven**，**int** 變數 **p**，**double** 變數 **s** (3%)
 - (b) 在螢幕顯示一行字，要求使用者輸入一個正整數 **p** (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 **p** (3%)
 - (d) 將檢查 **p** 是否為偶數的邏輯關係式，設定給已宣告之 **bool** 變數 **isEven** (3%)
 - (e) 寫一個 **if-else** 敘述，當已宣告之 **bool** 變數 **isEven** 為 **true** 時，在螢幕顯示"**p為偶數**"，否則顯示"**p為奇數**" (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 **n**，用遞增算子 **++** 加 **1** 後，設值給他處已宣告設值的 **int** 變數 **m** (3%)
 - (b) 令他處已宣告設值之 **int** 變數 **n** 及 **m**，計算 **n** 除以 **m** 的餘數，設定給他處已宣告之 **int** 變數 **r** (3%)
 - (c) 宣告**double**變數 **y**，並設定其值為數學函數 $\sin\left(\frac{2\pi}{\lambda}x\right)$ 。此處 λ 和 x 用 **double**變數 **lambda** 和 **x** 代表， π 則為圓周率。假定 **lambda** 和 **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 **x**，數值大於等於 **0** 時，設定變數 **y** 的數值為 **1**，反之則令 **y** 值為 **0**。假設 **y** 已在他處宣告 (3%)
 - (e) 宣告變數 **c** 為 **char** 型別，並令其值代表水平定位(tabulation)鍵 (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**，讓系統以時間及網路卡ID 決定其種子數 (3%)
- (b) 宣告兩個一維 **double** 陣列 **x** 與 **w**，其內容分別為 {0.2, 0.3, 0.1} 及 {0.5, 0.1, 0.2} (3%)
- (c) 宣告整數變數 **n**，並設初值為 **x** 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 **x[i]** 的乘積，否則不動作。迴圈結束後，以 **y** 為參數，呼叫 **static double** 函式 **Sigmoid**，將函式值設為 **double** 變數 **output** (3%)
- (e) 寫一個 **static double** 函式 **Sigmoid**，設其輸入參數為 **double y**，傳回算式 $\frac{1}{1+e^y}$ 之值。注意函數 e^y 可以呼叫 **Math.Exp(y)** 來計算 (3%)

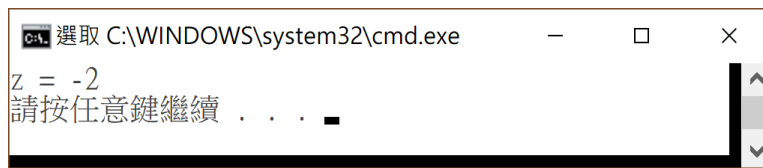
4. 找出以下程式片段之錯誤，並在盡量保持原先程式碼之前提下，予以更正。
 假設 **using System;** 敘述已經包含於程式中。

- (a) (3%) (一個語法錯誤) 執行時螢幕應顯示



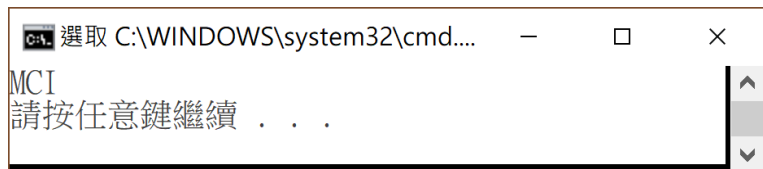
```
const int n = 1;
int sum = 0;
while (n <= 100)
{
    sum += n;
    ++n;
}
Console.WriteLine("sum = {0}", sum);
```

(b) (3%) (一個語義錯誤) 執行時螢幕應顯示



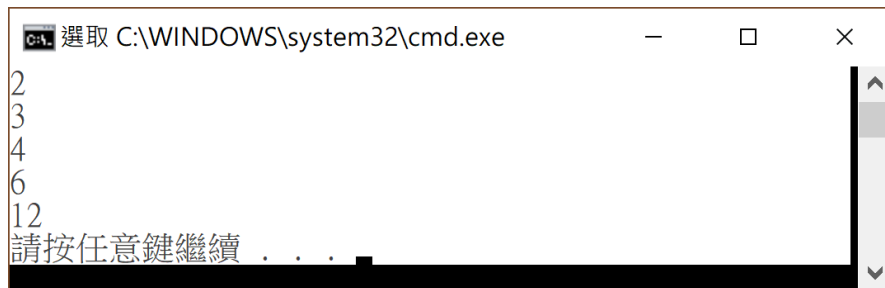
```
int a = 2;
int b = 2;
int c = -4;
int z = 0;
if (a == 0)
    if (b == 0) z = -1;
else z = c / b;
Console.WriteLine("z = {0}", z);
```

(c) (3%) (一個語法錯誤) 執行時螢幕應顯示



```
enum Diagnosis
{
    NORMAL,
    MCI
}
. . . . .
Diagnosis diagnosis = MCI;
switch(diagnosis)
{
    case Diagnosis.NORMAL:
        Console.WriteLine("Normal");
        break;
    case Diagnosis.MCI:
        Console.WriteLine("MCI");
        break;
    default:
        Console.WriteLine("Invalid diagnosis");
        break;
}
```

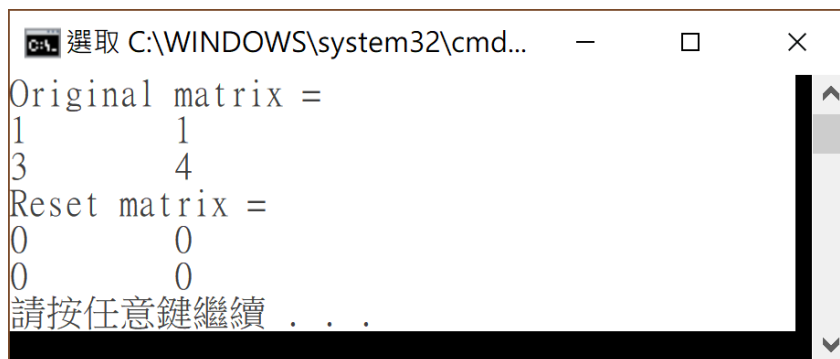
(d) (3%) (一個語義錯誤) 執行時螢幕應顯示所有 12 的因數如下:



```
C:\> 選取 C:\WINDOWS\system32\cmd.exe
2
3
4
6
12
請按任意鍵繼續 . . .
```

```
int x = 2;
int y = 12;
for(;;)
{
    if(y % x == 0)
    {
        Console.WriteLine(x);
    }
    ++x;
    if (x >= y) break;
}
```

(e) (3%) (一個語義錯誤) 執行時螢幕應顯示如下:



```
C:\> 選取 C:\WINDOWS\system32\cmd...
Original matrix =
1      1
3      4
Reset matrix =
0      0
0      0
請按任意鍵繼續 . . .
```

```
static void Main(string[] args)
{
    int[,] matrix = { {1, 1},
                      {3, 4} };

    int[,] copy = matrix;
    InverseAndReset(matrix);
    Console.WriteLine("Original matrix = ");
    Console.WriteLine(
```

```

        "{0} \t {1}", copy[0,0], copy[0,1]);
    Console.WriteLine(
        "{0} \t {1}", copy[1,0], copy[1,1]);
    Console.WriteLine("Reset matrix = ");
    Console.WriteLine(
        "{0} \t {1}", matrix[0, 0], matrix[0, 1]);
    Console.WriteLine(
        "{0} \t {1}", matrix[1, 0], matrix[1, 1]);
}

static void InverseAndReset(int[,] matrix)
{
    // inverse
    int[,] inverse = new int[2,2];
    int delta =
        matrix[0, 0] * matrix[1, 1] -
        matrix[0, 1] * matrix[1, 0];
    inverse[0, 0] = matrix[1, 1]/delta;
    inverse[1, 1] = matrix[0, 0]/delta;
    inverse[0, 1] = matrix[1, 0]/delta;
    inverse[1, 0] = matrix[0, 1]/delta;
    matrix = inverse;

    // reset
    matrix = new int[2, 2];
    matrix[0, 0] = 0;
    matrix[0, 1] = 0;
    matrix[1, 0] = 0;
    matrix[1, 1] = 0;
}

```

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 sum = 0.0;
        double term = 1.0;
        do
        {
            sum += term;

            // round off error can be ignored
            Console.WriteLine(
                "term = {0}, sum = {1}", term, sum);

            term *= 0.1;
        } while (Math.Abs(term) > 1.0e-4);
    }
}

```

6. 試寫出下列程式的螢幕輸出 (10 %)

```

using System;

namespace Problem6
{
    class Program
    {
        static void Main(string[] args)
        {
            int[,] source = { {1, 5, 7, 9, 3},
                               {1, 3, 8, 6, 2},
                               {2, 4, 8, 4, 1},
                               {0, 2, 6, 5, 0},
                               {1, 3, 9, 7, 2} };

            int[,] filter = { {1, 0, 0},
                               {0, 1, 0},
                               {0, 0, 1} };

            DisplayImage(source);
        }
    }
}

```

```

        int[,] resultC = ConvolveImage(source, filter);
        DisplayImage(resultC);
        int[,] resultM = MaxPoolImage(resultC, 2, 2);
        DisplayImage(resultM);
    }

```

```

static void DisplayImage(int[,] image)
{
    for(
        int i = 0; i < image.GetUpperBound(0)+1; ++i)
    {
        for(
            int j = 0; j < image.GetUpperBound(1)+1; ++j)
        {
            Console.Write("{0} ", image[i, j]);
        }
        Console.WriteLine();
    }
}

```

```

// assume stride = 1 (不用理會這個註解)
static int[,] ConvolveImage(
    int[,] source, int[,] filter)
{
    int nRowSource = source.GetUpperBound(0) + 1;
    int nColSource = source.GetUpperBound(1) + 1;
    int nRowFilter = filter.GetUpperBound(0) + 1;
    int nColFilter = filter.GetUpperBound(1) + 1;
    int nRowResult = nRowSource - nRowFilter + 1;
    int nColResult = nColSource - nColFilter + 1;
    int[,] result =
        new int[nRowResult, nColResult];

    // assume that stride = 1
    for(int i = 0; i < nRowResult; ++i)
    {
        for(int j = 0; j < nColResult; ++j)
        {

```

```

        result[i, j] =
            Convolve(i, j, source, filter);
    }
}
return result;
}

static int Convolve(
    int iResult, int jResult,
    int[,] image, int[,] filter)
{
    int nRowImage = image.GetUpperBound(0) + 1;
    int nRowTiling = filter.GetUpperBound(0) + 1;
    int maxIImage = iResult + nRowTiling - 1;
    if (maxIImage > nRowImage - 1) return 0;
    int nColTiling = filter.GetUpperBound(1) + 1;
    int nColImage = image.GetUpperBound(1) + 1;
    int maxJImage = jResult + nColTiling - 1;
    if (maxJImage > nColImage - 1) return 0;
    int r = 0;
    for(int i = 0; i < nRowTiling; ++i)
    {
        for(int j = 0; j < nColTiling; ++j)
        {
            r +=
                image[i+iResult, j+jResult] * filter[i, j];
        }
    }
    return r;
}

// assume stride = 1 (不用理會這個註解)
static int[,] MaxPoolImage(
    int[,] source, int nRowPool, int nColPool)
{
    int nRowSource = source.GetUpperBound(0) + 1;
    int nColSource = source.GetUpperBound(1) + 1;

```



```

        int nRowResult = nRowSource - nRowPool + 1;
        int nColResult = nColSource - nColPool + 1;
        int[,] result =
            new int[nRowResult, nColResult];

        for (int i = 0; i < nRowResult; ++i)
        {
            for (int j = 0; j < nColResult; ++j)
            {
                result[i, j] =
                    MaxInPool(i, j, source, nRowPool, nColPool);
            }
        }
        return result;
    }

    static int MaxInPool(int iResult, int jResult,
        int[,] image, int nRowPool, int nColPool)
    {
        int nRowImage = image.GetUpperBound(0) + 1;
        int maxIImage = iResult + nRowPool - 1;
        if (maxIImage > nRowImage - 1) return 0;
        int nColImage = image.GetUpperBound(1) + 1;
        int maxJImage = jResult + nColPool - 1;
        if (maxJImage > nColImage - 1) return 0;
        const int LARGE = 32767;
        int r = -LARGE;
        for (int i = 0; i < nRowPool; ++i)
        {
            for (int j = 0; j < nColPool; ++j)
            {
                r =
                    Math.Max(r, image[i+iResult, j+jResult]);
            }
        }
        return r;
    }
}

```

}

7. 台灣社會的老人化日趨嚴重，患有失智症(dementia)的老人也逐漸增加，成為老人醫療的重要課題。造成失智症的原因很多，但以罹患阿茲海默症(Alzheimer's Disease, AD)為大宗。AD 失智症的早期症狀，通常稱為輕度知能障礙(Mild Cognitive Impairment, MCI)。雖然目前對於失智症以及 MCI，並沒有有效的醫療方式，及早發現確診 MCI，還是有許多好處，例如：研究評估失智症進展期程、有效診斷及處置規劃、讓病人參與新藥研發、評估影響失智症病情因素等。(參閱 D. Kokkinakis, K. L. Fors, E. Björkner, and A. Nordlund, "Data Collection from Persons with Mild Forms of Cognitive Impairment and Healthy Controls - Infrastructure for Classification and Prediction of Dementia," *Proceedings of the 21st Nordic Conference of Computational Linguistics*, pages 172–182, Gothenburg, Sweden, 2017.)

觀察 AD 失智症患者的日常生活，可以發現他們的記憶及語言表達能力較差；因此蒐集患者(實驗組)及正常老人(對照組)話語，建立語料庫(corpus)，抽取其語言特徵，以便利用統計及機器學習，由語言特徵篩選 MCI。這種方法簡便、非侵入性、低成本，因此成為當前 MCI 檢測的重要研究方向。

目前所知的華語及台語的失智症篩檢語料庫，僅有國立清華大學呂菁菁教授(<http://gitll.web.nthu.edu.tw/files/15-1998-22778,c2-1.php?Lang=zh-tw>) 所建語料庫，收錄於包含數種語言的失智症篩檢語料庫 DementiaBank (<https://talkbank.org/DementiaBank/>)之中。

呂教授的語料庫，使用國際學界，評估受試者認知及語言能力的「偷餅乾」測試(query with cookie theft)蒐集語料，方法如下：

受測者看圖 1「偷餅乾」圖形，口語描述圖片中發生的事情，越詳細越好。



圖 1. 「偷餅乾」圖。取自 H. Goodglass and E. Kaplan, *The Assessment of Aphasia and Related Disorders*, Lea&Febiger, USA, 1983.

受試者的話語經過 Speech-To-Text 處理，轉換為文本。一般來說，受測者的描述大約 7 到 10 句，或更多。呂教授語料庫內的一位失智受試者，對於圖形的部分敘述為(台大電機系碩士生劉昱佑研究進度報告，2018 年 2 月 24 日):

「他站在椅子上拿東西，這水龍頭放的水在洗東西，洗窗戶，洗碗，他站在椅子上拿東西，凳子，水龍頭。」

受試者想表達的應該是:

「男孩站在椅子上拿東西，媽媽用水龍頭的水洗碗。」

另一位失智患者的部分描述為:

「樓上拿東西，到樓上拿東西，婦女在洗東西，到樓上拿餅乾，水池裏面洗碗」

他的意思或許是:

「男孩到櫥櫃上拿餅乾，婦女在水池裡面洗碗。」

從這些句子看來，顯然失智症患者的語言表達與正常人不同。但是，怎樣不同呢？從計算語言學(computational linguistics)的角度看來，如果能夠擷取出有效的統計量特徵，其大小分布情形與距離，便可分辨是否為失智患者。

一篇美國的碩士論文(A. Habash, *Language Analysis of Speakers with Dementia of the Alzheimer's Type*, Master Thesis, University of North Carolina Wilmington, 2012.) 提到，失智老人日常英文對話之文句特徵，可以分為句型(syntax)、句義(semantics)、實效(pragmatic)三大類，總數達數百種，但各有優劣。除了文句特徵外，語音的聲學特徵，如語調或說話停頓比例等，當然也可以有效應用。但為簡化問題，此處略去其相關討論。

假定失智老人日常對話(dialogue)的大部分特徵，也可以用來分析看圖說故事的獨白(monologue)；我們便可以計算其特徵，再以統計檢定失智者與正常人言語特徵差異的顯著性。利用這些特徵差異，搭配機器學習，就能讓電腦自動偵測出 MCI 的患者。

在眾多的文句特徵中，代表某受測者的句型特徵包括名詞出現率(Noun Rate, NR)、代名詞出現率(Pronoun Rate, PR)、動詞出現率(Verb Rate, VR)等。其計算公式分別為:

$$NR = N/W$$

$$PR = P/W$$

$$VR = V/W$$

其中 N、P、V、W 分別代表整個語料庫中，受試者名詞、代名詞、動詞、全部字詞的數量。雖然原先的論文使用英文語料，這三種特徵在中文語料庫，應該也是可以應用的特徵。

以前面提出來的例子而言，假設語料庫 MCI 患者 A 的語料，經由中文斷詞系統，分割出各個單字或字詞，得到：

他 站 在 椅 子 上 拿 東 西 這 水 龍 頭 放 的 水 在 洗 東 西
洗 窗 戶 洗 碗 他 站 在 椅 子 上 拿 東 西 凳 子 水 龍 頭

顯然字詞總數 $W=28$ ，名詞數 $N=11$ (椅子、東西、水龍頭、水、東西、窗戶、碗、椅子、東西、凳子、水龍頭)，代名詞數 $P=3$ (他、這、他)，動詞數 $V=8$ (站、拿、放、洗、洗、洗、站、拿)。對應的三種特徵為

$$NR = 11/28 \approx 0.392857$$

$$PR = 3/28 \approx 0.107143$$

$$VR = 8/28 \approx 0.285714$$

同時，假定正常人 B 的語料字詞為:

男 孩 站 在 椅 子 上 拿 東 西 媽 媽 用 水 龍 頭 的 水 洗 碗

則 $W=13$ ， $N=7$ (男孩、椅子、東西、媽媽、水龍頭、水、碗)， $P=0$ ， $V=4$ (站、拿、用、洗)。對應特徵數值

$$NR = 7/13 \approx 0.538462$$

$$PR = 0$$

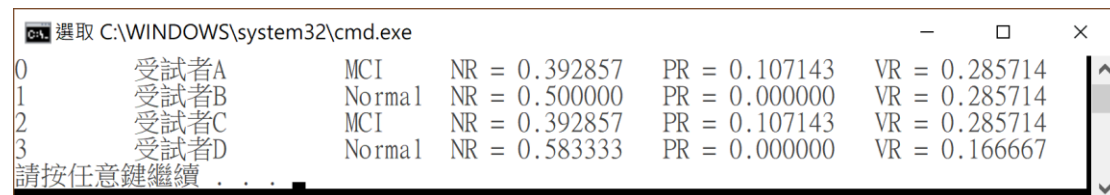
$$VR = 4/13 \approx 0.307692$$

可以看出 A、B 兩人的語句特徵 NR 相差較大，PR 及 VR 差距較不明顯。雖然此處的語料庫並不實際，多少可以說明以語言特徵篩選 MCI 患者的概念。

本題請你依照如下要求，撰寫一個C#程式。

1). 程式功能：

計算並顯示語料庫中，每位受試者的流水號、代號、診斷、NR、PR、VR。程式輸出如圖 2。



```
C:\> 選取 C:\WINDOWS\system32\cmd.exe
0 受試者A MCI NR = 0.392857 PR = 0.107143 VR = 0.285714
1 受試者B Normal NR = 0.500000 PR = 0.000000 VR = 0.285714
2 受試者C MCI NR = 0.392857 PR = 0.107143 VR = 0.285714
3 受試者D Normal NR = 0.583333 PR = 0.000000 VR = 0.166667
請按任意鍵繼續 . . . .
```

圖 2. 程式輸出螢幕畫面。

2). 背景：

假設語料庫及名詞、代名詞、動詞詞典分別已建立為二維矩形和一維陣列，宣告於主程式開頭的部分。同時假定某受測者的語料結束後，以"X"填補不足的部分。如以下程式片段所示：

```
namespace Problem7
{
    public class Program
    {
        static void Main(string[] args)
        {
            string[,] corpus = {
                {"受試者A", "MCI",
                "他", "站", "在", "椅子", "上", "拿", "東西", "這",
                "水龍頭", "放", "的", "水", "在", "洗", "東西", "洗",
                "窗戶", "洗", "碗", "他", "站", "在", "椅子", "上",
                "拿", "東西", "凳子", "水龍頭"},
                {"受試者B", "Normal",
                "男孩", "站", "在", "椅子", "上", "拿", "東西", "媽媽",
                "用", "水龍頭", "的", "水", "洗", "碗", "X", "X",
                "X", "X", "X", "X", "X", "X", "X", "X",
                "X", "X", "X", "X"},
                {"受試者C", "MCI",
                "他", "站", "在", "椅子", "上", "拿", "東西", "這",
                "水龍頭", "放", "的", "水", "在", "洗", "東西", "洗",
                "窗戶", "洗", "碗", "他", "站", "在", "椅子", "上",
```

