通識計算機程式設計期中考參考解答

臺灣大學 鄭士康 4/18/2014

1.

(a)
```
int n;
bool b;
double z;
```
(3%)

(b)
```
Console.WriteLine("輸入一個浮點數: ");
```
(3%)

(c)
```
z = double.Parse(Console.ReadLine());
```
(3%)

(d)
```
n = (int)z;
```
(3%)

(e)
```
b = (n >= 0 && n <= 100);
```
(3%)

2.

(a)
```
m = ++n;
```
(3%)

(b)
```
r = t % 7;
```
(3%)

(c)
```
double d = Math.Sqrt(x * x + y * y + z * z);
```
(3%)

(d)
```
int v = ( u > -10.0 )? 1 : 0;
```
(3%)

(e)
```
char c = '\'';
```
(3%)

3.

(a)
```
Random rand = new Random(360);
int k = rand.Next();
```
(3%)

(b)
```
const double RESTING_VOLTAGE = -70.0;
```
(3%)

(c)
```
int score = 0;
do
{
    Console.Write("輸入一個0(含)與100(含)之間的數: ");
    score = int.Parse(Console.ReadLine());
}
while (score > 100 || score < 0);
```
(3%)

(d)
```
Console.Write("輸入一個字串");
char[] s = Console.ReadLine().ToCharArray();
int idx = Array.IndexOf(s, 'g');
```
(3%)

(e)
```
static void f(ref int x)
{
```

```
        x++;
    }
```

(3%)

4. 找出以下程式片段之錯誤，並予更正。

(a) (3%) 附註符號「**//**」效力僅有一行，因此第二行敘述之說明文字不能算是附註，可改為:

```
string info = ""; // 初值設為空字串,
                  // 之後會改設定為另外讀入的文字資訊
```

(b) (3%) 比較兩數時之關係算符為「==」，不是「=」(代表設定， assignment). 應改為

```
int n = 0;
while(n < 100)
{
   if(n == 10) continue;
   n++;
}
```

(c) (3%) **x** 與 **y** 均為整數時, **x/y** 代表其商，捨去了小數部分，所以應先將**x** 或 **y** 強迫轉型為 **double.** 修改如下:

```
int x = 3;
int y = 7;
double ratio = (double) x/y;
```

(d) (3%) **for** 迴圈由 **n = 1** 開始，到 **n = 3** 時，尚未執行迴圈中之敘述即已跳出，所以可改為由 **n = 0** 開始即可. 其他簡單之改法只要能達到執行回圈內敘述 3 次也可以.

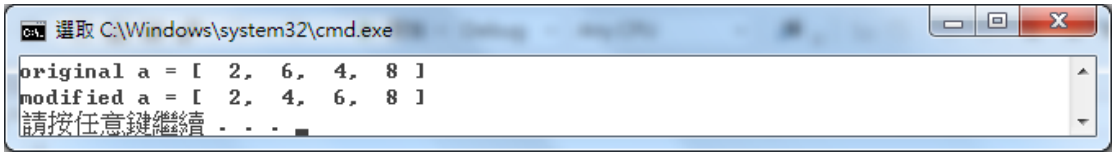
```
for(int n = 0; n < 3; n++)
{
  Console.WriteLine("通識程設老師很機車!");
}
```

(e) (3%) 呼叫函式**ModifyArray**時僅複製陣列**a** 之參考(reference) 傳給函式**Modify**, 所以在函式中的 **a = new int[] { 7, 8, 9 }** 敘述僅將陣列**a**參考的copy改為新陣列的參考, 並不影響**Main**程式中的**a**陣列. 所以改成傳址參數及呼叫即可:

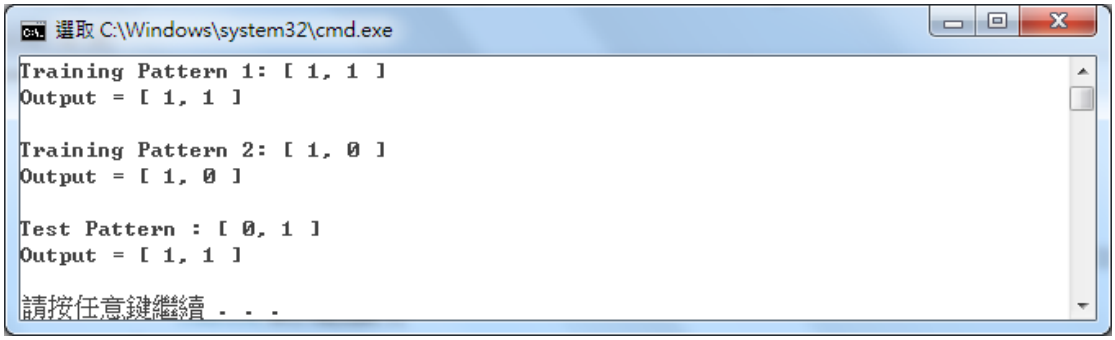
```
class Program
{
  static void Main(string[] args)
  {
      int[] a = { 1, 2, 3, 4, 5 };
      ModifyArray(ref a);
  }

  static void ModifyArray(ref int[] a)
  {
      a = new int[] { 7, 8, 9 };
  }
}
```

5. (5 %) 所得螢幕輸出如下:



6. (10 %) 所得螢幕輸出如下:



7. (25%)
這個題目出題時考慮不夠週到, 讓雙方都採取「寬容的以牙還牙」策略, 一開始兩邊都「合作」(假設對方是好人), 接著以牙還牙, 採取對方上一次的行動,

結果就一直合作下去了(大概培養出了很高的默契，知道對方可信賴). 雖然題目這樣的設定產生單調的結果，我們還是要照問題背後的邏輯寫程式，例如，將「寬容的以牙還牙」寫成一個函式，當題目要求 B 改採其他策略，例如，完全不管上次結果，隨機決定合作或背叛，就只要補一個函式，做這種決策，再在主程式 B 做決定時，改成呼叫新函式就可以了，程式其他部分都不必修改，這才是這個問題的精神.

以下依照這個想法寫成的程式可供參考

```
/*
 * 雙方均採用"寬恕地以牙還牙"策略的"重複囚徒困境"模擬
 * Shyh-Kang Jeng, 4/16/2014
 *
 * 虛擬碼
 * 1. 讀入兩個整數，分別作為A, B決策所需亂數產生器的種子數
 * 1. 以不用種子數的亂數產生器產生一個1到1000的亂數N
 * 2. 設定A, B決策的初值均為"合作"
 * 3. 印出A, B決策及各自得分
 * 4  for(i = 1; i< N; i++)
 *    {
 * 4.1  將上次A, B的決策存起來
 * 4.2  以A的亂數產生器產生一個0到99的亂數
 * 4.3  決定A採用之決策
 * 4.4  以B的亂數產生器產生一個0到99的亂數
 * 4.5  決定B採用之決策
 * 4.6  計算雙方得分
 * 4.7  累計雙方得分
 * 4.8  顯示雙方決策及累計得分
 *    }
 *
 * 函式: 採用之決策(上次對手決策, 0到99的亂數,
 *       寬恕對方機率之百分數)
 * 1. if(上次對手決策為"合作")
 *    {
 * 1.1     決策設為"合作"
 *    }
 *    else if( 0到99的亂數 < 寬恕對方機率之百分數 )
```

```
 *    {
 * 1.2     寬恕，決策為"合作"
 *    }
 *    else
 *    {
 * 1.3     報復，決策為"背叛"
 *    }
 * 2. 傳回決策
 *
 *
 * 函式：計算雙方得分(A決策， B決策，算出之A得分，算出之B得分)
 * 1. if(A決策為"合作" 且 B決策為"合作")
 *    {
 * 1.1   A得分為3
 * 1.2   B得分為3
 *    }
 *    else if(A決策為"背叛" 且 B決策為"合作")
 *    {
 * 1.3   A得分為5
 * 1.4   B得分為0
 *    }
 *    else if(A決策為"合作" 且 B決策為"背叛")
 *    {
 * 1.5   A得分為0
 * 1.6   B得分為5
 *    }
 *    else
 *    {
 * 1.7   A得分為1
 * 1.8   B得分為1
 *    }
 *
 */

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace Problem7
{
    enum Decision
    {
        COOPERATE,
        BETRAY
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.Write(
                "輸入一個整數作為A決策所需亂數產生器之種子數: ");
            int seed_A = int.Parse(Console.ReadLine());
            Random rand_A = new Random(seed_A);
            Console.Write(
                "輸入一個整數作為B決策所需亂數產生器之種子數: ");
            int seed_B = int.Parse(Console.ReadLine());
            Random rand_B = new Random(seed_B);
            Random rand = new Random();
            //int N = rand.Next() % 1000 + 1;
            int N = 5;
            Decision decision_A = Decision.COOPERATE;
            Decision decision_B = Decision.COOPERATE;
            int score_A;
            int score_B;
            ComputeScores( decision_A, decision_B,
                out score_A, out score_B );
            int scoreSum_A = score_A;
            int scoreSum_B = score_B;
            DisplayDecisionsAndAccumulatedScores(0,
                decision_A, decision_B, scoreSum_A,
                scoreSum_B);
            int i;
            Decision lastDecision_A;
```

```
    Decision lastDecision_B;
    int n_A = 0;
    int n_B = 0;
    for(i = 1; i <= N; i++)
    {
        lastDecision_A = decision_A;
        lastDecision_B = decision_B;
        n_A = rand_A.Next() % 100;
        n_B = rand_B.Next() % 100;
        decision_A = DecisionTaken( lastDecision_B,
            n_A, 5 );
        decision_B = DecisionTaken( lastDecision_A,
            n_B, 2 );
        ComputeScores( decision_A, decision_B,
            out score_A, out score_B );
        scoreSum_A += score_A;
        scoreSum_B += score_B;
        DisplayDecisionsAndAccumulatedScores(i,
            decision_A, decision_B, scoreSum_A,
            scoreSum_B);
    }
}

static Decision DecisionTaken(
    Decision lastOpponentDecision, int n,
    int forgivenessPercentage )
{
    Decision action;
    if( lastOpponentDecision == Decision.COOPERATE )
    {
        action = Decision.COOPERATE;
    }
    else if( n < forgivenessPercentage )
    {
        action = Decision.COOPERATE;
    }
    else
    {
```

```
        action = Decision.BETRAY;
    }
    return action;
}

static void ComputeScores( Decision decision_A,
    Decision decision_B, out int score_A,
    out int score_B )
{
    if(decision_A == Decision.COOPERATE &&
        decision_B == Decision.COOPERATE)
    {
        score_A = 3;
        score_B = 3;
    }
    else if(decision_A == Decision.BETRAY &&
        decision_B == Decision.COOPERATE)
    {
        score_A = 5;
        score_B = 0;
    }
    else if(decision_A == Decision.COOPERATE &&
        decision_B == Decision.BETRAY)
    {
        score_A = 0;
        score_B = 5;
    }
    else
    {
        score_A = 1;
        score_B = 1;
    }
}

static void DisplayDecisionsAndAccumulatedScores(
    int i, Decision decision_A, Decision decision_B,
    int scoreSum_A, int scoreSum_B)
{
```

```
            Console.Write("回合 {0}: A ", i);
            DisplayDecision(decision_A);
            Console.Write(", 累積分數 {0};", scoreSum_A);
            Console.Write("\t B ");
            DisplayDecision(decision_B);
            Console.Write(", 累積分數 {0}", scoreSum_B);
            Console.WriteLine();
        }

        static void DisplayDecision(Decision decision)
        {
            switch (decision)
            {
                case Decision.COOPERATE:
                    Console.Write("合作");
                    break;
                case Decision.BETRAY:
                    Console.Write("背叛");
                    break;
                default:
                    Console.Write("不應到此");
                    break;
            }
        }
    }
}
```