通識計算機程式設計期中考參考解答

臺灣大學 鄭士康 11/11/2016

1.

(a) (3%)

```
bool valid;
byte xb;
int xi;
```

(b) (3%)

```
Console.WriteLine("輸入一個0到255之間的整數");
```

(c) (3%)

```
xi = int.Parse(Console.ReadLine());
```

(d) (3%)

```
valid = (xi >= 0 && xi < 256);
```

(e) (3%)

```
if (valid)
{
   xb = (byte)xi;
}
else
{
   xb = 0;
}
```

2.

(a) (3%)

```
--m;
```

(b) (3%)

```
dx = x1 - x2;
```

(c) (3%)

```
double distance = Math.Sqrt(dx * dx + dy * dy);
```

(d) (3%)

```
double y = (x >= 0) ? x: 0.0;
```

(e) (3%)

```
char c = '\'';
```

3.

(a) (3%)

```
Random rand = new Random();
```

(b) (3%)

```
const int N = 10;
bool[] labels = new bool[N];
```

(c) (3%)

```
for(int i = 0; i < N; ++i)
{
  labels[i] = (rand.Next() % 2 == 1);
}
```

(d) (3%)

```
double[] distances = new double[N];
for(int i = 0; i < N; ++i)
{
  distances[i] = rand.NextDouble();
}
```

(e) (3%)

```
static bool LabelOfNearestElement(
  double[] distances, bool[] labels)
{
  Array.Sort(distances, labels);
  return labels[0];
}
```

4.

(a) (3%) (一個語義錯誤)

敘述 **Write** 不跳行，使使用者輸入的學號仍在同一行。

```
Console.WriteLine("請輸入學號: ");
string registerNumber = Console.ReadLine();
```

(b) (3%) (一個語義錯誤)

敘述 **z = y++;** 中，先將 **y** 設值給 **z** 之後，才將 **y** 遞增為 **5**，所

以輸出的 **z** 為 **4**，而不是想要的 **5**。

```
int y = 4;
int z = ++y;
Console.WriteLine("{0} 是 y 增加 1", z);
```

(c) (3%) (一個語義錯誤)

迴圈控制變數 **n** 在迴圈內沒有改變，成為無窮迴圈。

```
int f_n_minus2 = 1;
int f_n_minus1 = 2;
int f;
int n = 1;
do
{
   f = f_n_minus1 + f_n_minus2;
   Console.WriteLine("F({0}) = {1}", n, f);
   f_n_minus2 = f_n_minus1;
   f_n_minus1 = f;
   ++n;
 } while(n <= 4);
```

(d) (3%) (一個語義錯誤)

table.GetUpperBound(0) 代表 **table** 第一個索引的最大可能值，等於 **3**，因此迴圈控制變數 **i** 最多只能增加到 **2**，少執行一次。

```
int[,] table = new int[4, 2];
for (int i = 0; i < table.GetUpperBound(0)+1; ++i)
{
   for(int j = 0; j <= 1; ++j)
   {
      table[i, j] = i + j;
   }
   Console.WriteLine(
     table[i, 0] + "\t" + table[i, 1]);
}
```

(e) (3%) (一個語義錯誤)

呼叫 **F00** 為傳值呼叫，將 **a** 陣列的參考，複製一份給函式 **F00** 的形式參數。在 **F00** 中，**a** 陣列的每個元素加 **2** 之後，另外向作業系統要求記憶空間給新陣列 **{ 1, 6, 8 }** ，並將新陣列參考設定給 **F00** 的形式參數。然而，**Main** 函式中的 **a** 陣列參考並沒有受影響，因此仍然輸出

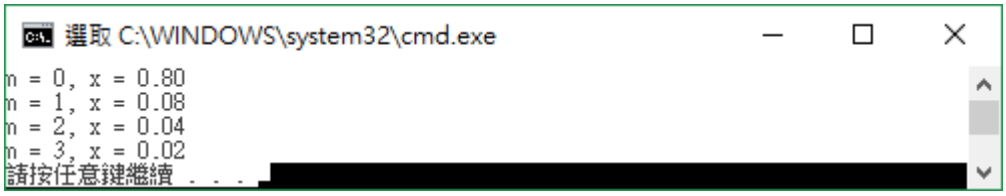
```
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
```

若要螢幕輸出新陣列 **{1, 6, 8}**，必須讓 **F00** 直接改變 **a** 陣列參考，所以要改採傳址呼叫。

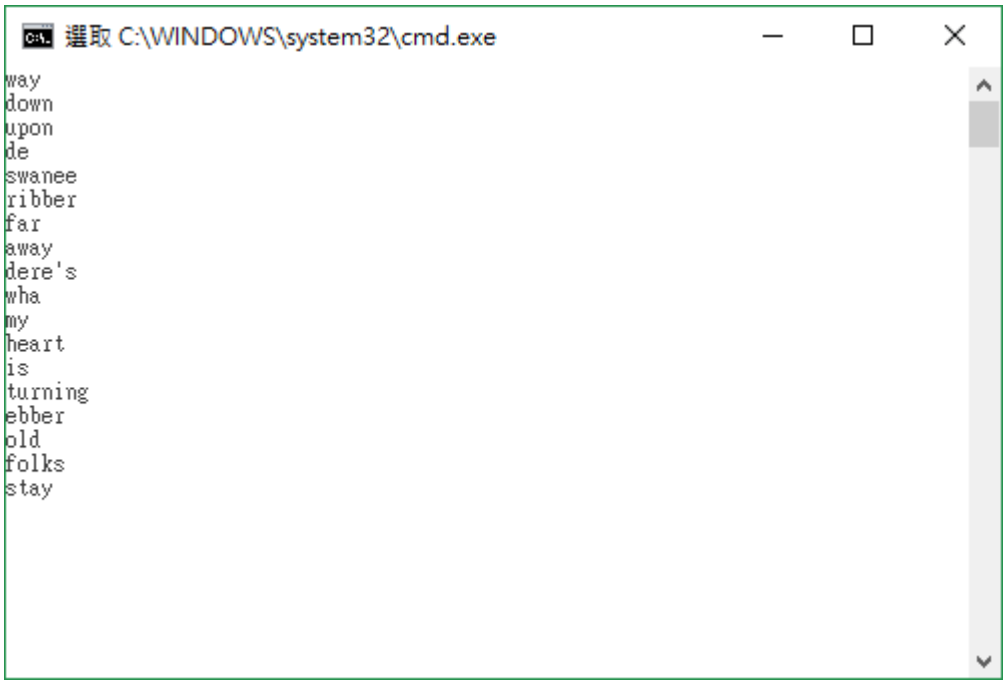
```
static void Main(string[] args)
{
  int[] a = {1, 2, 3, 4, 5};
  F00(ref a);
  for(int i = 0; i < a.Length; ++i)
  {
     Console.WriteLine("a[" + i + "] = " + a[i]);
  }
}

static void F00(ref int[] a)
{
   for(int i = 0; i < a.Length; ++i)
   {
      a[i] += 2;
   }
   a = new int[3] {1, 6, 8};
 }
```

5. (5 %)



6. (10 %)



7. (25%)

```
using System;

namespace Problem7
{
  public class Program
  {
     static void Main(string[] args)
     {
        // set up training data
        double[,] trainingData;
        bool[] trainingLabels;
        GenerateData(out trainingData, out trainingLabels);
```

```
  // input a test data
  double[] testData;
  int k;
  Input(out k, out testData);

  // find nearest k neighbors and output the label
  bool testLabel = LabelBy_k_NearestNeighbors(k,
    testData, trainingData, trainingLabels);
  Output(testData, testLabel);
}

public static bool LabelBy_k_NearestNeighbors(int k,
  double[] testData, double[,] trainingData,
  bool[] trainingLabels )
{
  int nTrainingData = trainingData.GetUpperBound(0)+1;
  int[] numbers = new int[nTrainingData];
  double[] distances = new double[nTrainingData];
  for (int i = 0; i < nTrainingData; ++i)
  {
     numbers[i] = i;
     distances[i] = Distance(testData[0], testData[1],
       trainingData[i, 0], trainingData[i, 1]);
  }
  Array.Sort(distances, numbers);
  bool result = LabelByCounting_k_NearestLabels(k,
    numbers, trainingLabels);
    return result;
}

static bool LabelByCounting_k_NearestLabels(int k,
  int[] numbers, bool[] trainingLabels)
{
  int nTrue = 0;
  int nFalse = 0;
  for(int i = 0; i < k; ++i)
  {
     if (trainingLabels[numbers[i]])
```

```
    {
      ++nTrue;
    }
    else
    {
      ++nFalse;
    }
  }
  bool result = (nTrue >= nFalse);
  return result;
}

static double Distance(double x1, double y1,
  double x2, double y2)
{
  double dx = x2 - x1;
  double dy = y2 - y1;
  double result = Math.Sqrt(dx * dx + dy * dy);
  return result;
}

static void GenerateData(out double[,] trainingData,
  out bool[] trainingLabels)
{
  Console.Write("輸入訓練資料個數: ");
  int nTrainingData = int.Parse(Console.ReadLine());
  trainingData = new double[nTrainingData, 2];
  trainingLabels = new bool[nTrainingData];
  Random rand = new Random();

  Console.WriteLine("產生的訓練資料");
  Console.WriteLine(
    " 編號 \t x \t \t \t y \t \t \t 標籤");
  for(int i = 0; i < nTrainingData; ++i)
  {
    trainingData[i, 0] = rand.NextDouble();
    trainingData[i, 1] = rand.NextDouble();
    trainingLabels[i] = (rand.Next() % 2 == 1);
```

```
        Console.WriteLine("{0} \t {1} \t {2} \t {3}",
          i, trainingData[i, 0], trainingData[i, 1],
          trainingLabels[i]);
      }
    }

    static void Input(out int k, out double[] testData)
    {
      Console.Write("輸入 k: ");
      k = int.Parse(Console.ReadLine());
      Console.WriteLine(
        "輸入測試資料的 x, y 特徵數值, 以逗點分隔");
      string[] data = Console.ReadLine().Split(',');
      testData = new double[2];
      testData[0] = double.Parse(data[0]);
      testData[1] = double.Parse(data[1]);
    }

    static void Output(double[] testData, bool testLabel)
    {
      Console.Write("Test  data  =  [  {0},  {1}  ]",
        testData[0], testData[1]);
      Console.WriteLine("\t Label = " + testLabel);
    }
  }
}
```