通識計算機程式設計期中考參考解答

4/24/2020

1.

(a) (3%)

答:

```
bool  success;
int m;
double r;
```

(b) (3%)

答:

```
Console.Write("請輸入一個有小數點的實數: ");
```

或

```
Console.WriteLine("請輸入一個有小數點的實數: ");
```

(c) (3%)

答:

```
r = double.Parse(Console.ReadLine());
```

或

```
r = Convert.ToDouble(Console.ReadLine());
```

(d) (3%)

答:

```
m = (int) (r + 0.5);
```

(e) (3%)

答:

```
success = (m == 1);
```

2.

。

(a) (3%)

答:

```
factorial *= n;
```

(b) (3%)
答:

```
double theta = Math.PI/2;
double ct = Math.Cos(theta);
```

(c) (3%)
答:

```
double threshold = -80.0;
response = (v > threshold)? 1.0 : 0.0;
```

(d) (3%)
答:

```
string s1 = "MidTerm";
string s2 = "2020";
string testName = s1 + s2;
```

(e) (3%)
答:

```
char lf = '\n';
```

3.

(a) (3%)
答:

```
Random rand = new Random(777);
```

(b) (3%)
答:

```
const int LIST_SIZE = 100;
int[] grades = new int[LIST_SIZE];
```

(c) (3 %)
答:

```
static int NewGrade(Random rand)
{
  int grade = rand.Next(100);
  return grade;
}
```

(d) (3%)
答:

```
for(int i = 0; i < LIST_SIZE; ++i)
{
   grades[i] = NewGrade(rand);
}
```

(e) (3%)
答:

```
Array.Sort(grades);
Array.Reverse(grades);
topScore = grades[0];
Console.WriteLine("Top score = " + topScore);
```

4. 指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。

(a) (3%) (一個語義錯誤) 執行時螢幕應顯示

答:

由於 **s < t** 不成立，**t > s++** 不管是否成立，都無法改變整體測試

s < t 且 t > s++
為偽的結果. 因此使用邏輯”且”的運算符號 **&&** 時，
之後的條件測試 **t > s++**
會被跳過，使判斷條件的副作用 **s++** 不會被執行.
因此螢幕輸出變成
s = 5, t = 5
與要求不同。

解決的方法是改用 **&** 算符,
強迫每一個測試條件都要檢驗,
s++ 就會被執行.

程式修改如下(黃色底代表刪除):

```
int s = 5;
int t = 5;
if(s < t && & t > s++)
{
  t = 3;
}
```

```
else
{
  t = 5;
}
Console.WriteLine("s = {0}, t = {1}", s, t);
```

(b) (3%) (一個語義錯誤) 執行時螢幕應顯示

或

使用 Console.ReadLine() 時，會得到鍵盤按"Enter"或"Return"前，輸入的字串，不是可以計算的數值，計算機0與1記憶儲存的解讀方式與整數不同。因此需要把得到的字串 Parse (剖析) 或 Convert (轉換) 為可以計算的數值，儲存為可做整數解讀的0與1樣式。

程式修改如下:

```
Console.Write("輸入成績: ");
int grade = Console.ReadLine();
int grade = int.Parse(Console.ReadLine());
// 或 int grade = Convert.ToInt16(Console.ReadLine());
if(grade > 60)
    Console.WriteLine("成績為: " + grade);
else
  Console.WriteLine("成績不及格");
```

(c) (3%) (一個語義錯誤) 執行時螢幕應顯示

由所需輸出及程式碼排列(layout)來看，

```
else
  v = -1;
```

應該是 **if(u >= 5)** 不成立時，就應接著執行的敘述。
但是原先的寫法會使

```
else
  v = -1;
```

因為最靠近條件測試

```
  if( u >= 7 )
```

而成為 **if(u >= 7)** 不成立時，所執行的下一個敘述。
這是程式語言之懸置(dangling) **else** 問題的 C# 解法)。
解決這種問題的最佳辦法，就是善用大括弧，明確分出各個 **if** 及 **else** 的處理範圍(即使範圍中只有一行敘述)。所以原程式可以修改如下:

```
int u = 3;
int v = 0;
if ( u >= 5 )
{
  if( u >= 7 )
  {
    v = 1;
  }
}
else
{
  v = -1;
}
Console.WriteLine("v =" + v);
```

(d) (3%) (一個語義錯誤) 執行時螢幕應顯示如下:

do-while 迴圈的繼續條件 **b <= 0.25** 在第一次 iteration 時就無法滿足，因此最後 **b = 0.5, sum = 1**，與所要求不同.
題目的用意似乎是在計算等比級數

$$\text{sum} = 1 + \frac{1}{2} + \frac{1}{4}$$

其中的項越來越小。所以迴圈繼續條件應該是 **b** 不夠小 (大於某值)，才繼續累加。因此原程式修改如下:

```
double b = 1.0;
double sum = 0.0;
do {
    sum += b;
```

```
    b /= 2;
 } while (b <= > 0.25);
Console.WriteLine("sum =" + sum);
```

(e) (3%) (一個語義錯誤) 執行時螢幕應顯示如下:

此處 **Increment** 為傳值(pass by value)呼叫的函式，因此執行 **Increment(x)**時，會將主程式的變數 **x** 儲存的數值 **6**，複製一份，作為函式 **Increment** 的輸入參數(或稱引數)。函式中將 **x** 加一變成 **7**。但是這裡被加一的，是由主程式複製過來的**x**，**return** 它的值，不會影響到原程式中的變數 **x** (數值仍為 6)。

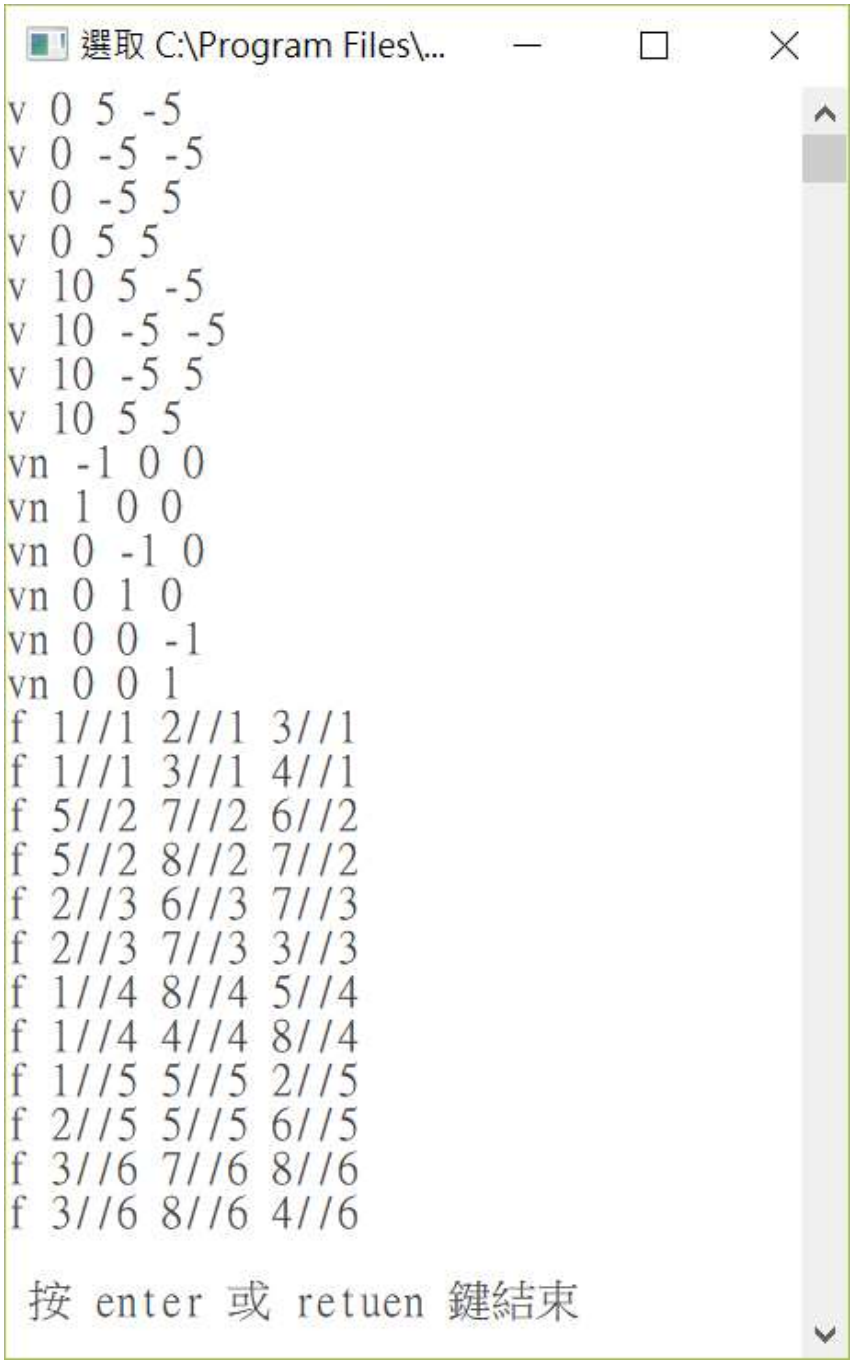
要得到希望的輸出(主程式的 **x** 有被加一)，最好改用傳址(pass by address)呼叫。原程式修改如下:

```
static void Increment(ref int x)
{
    x++;
}

static void Main(string[] args)
{
    int x = 6;
    Increment(ref x);
    Console.WriteLine("x = " + x);
}
```

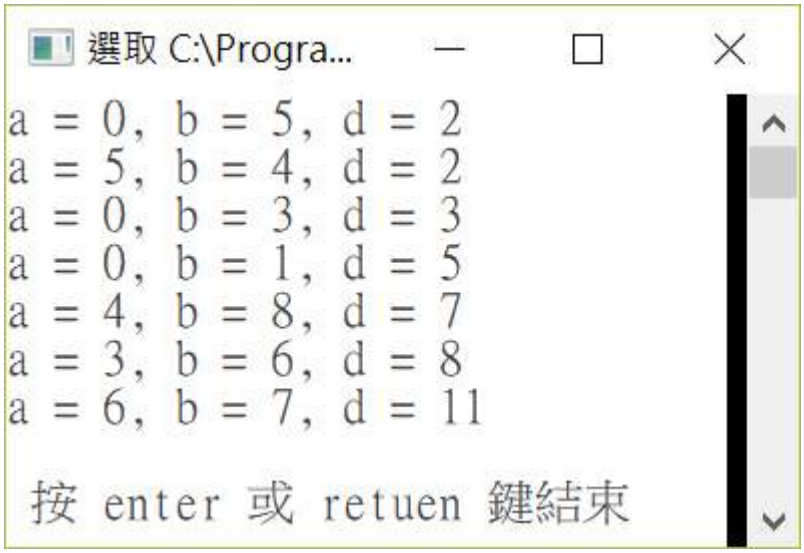
5. (5 %)

答:


選取 C:\Program Files\...
v 0 5 -5
v 0 -5 -5
v 0 -5 5
v 0 5 5
v 10 5 -5
v 10 -5 -5
v 10 -5 5
v 10 5 5
vn -1 0 0
vn 1 0 0
vn 0 -1 0
vn 0 1 0
vn 0 0 -1
vn 0 0 1
f 1//1 2//1 3//1
f 1//1 3//1 4//1
f 5//2 7//2 6//2
f 5//2 8//2 7//2
f 2//3 6//3 7//3
f 2//3 7//3 3//3
f 1//4 8//4 5//4
f 1//4 4//4 8//4
f 1//5 5//5 2//5
f 2//5 5//5 6//5
f 3//6 7//6 8//6
f 3//6 8//6 4//6
按 enter 或 retuen 鍵結束

6. (10 %)

答:



7. (25%)

答:

```
/*
* Program to simulate simplified diagnosis process
* for COVID 19.
*
* The process is based on a given binary decision tree
*
*                 檢驗條件
*              /           \
*           Y/              \N
*           /               \
*        確診病例           臨床條件
*                 /                \
*              Y /                  \N
*                /                   \
*               /                     \
*             /                        \
*          流行病學條件              流行病學條件
*          /          \             /          \
*        Y/            \N         Y/            \N
*        /              \         /              \
*    極可能病例      高度懷疑之   需自主健康管理      正常或無症狀
```

```
 *                 社區性肺炎    或隔離對象          之感染者
 *                 患者
 *
 * For definitions of 檢驗條件, 臨床條件, 流行病學條件, etc.,
 * see https://www.msn.com/zh-tw/news/living/
 *
%E6%8B%92%E7%B5%95%E6%BC%8F%E7%B6%B2%E4%B9%8B%E9%AD%9A%EF%BC%81%E6%8C
%87
 *
%E6%8F%AE%E4%B8%AD%E5%BF%83%E6%93%B4%E5%A4%A7%E3%80%8C%E6%96%B0%E5%86
%A0
 *
%E8%82%BA%E7%82%8E%E3%80%8D%E6%8E%A1%E6%AA%A2%E6%A8%99%E6%BA%96/ar-
BB11ZXrx
 *
 * The above decision tree is implemented by one-dimensional arrays
 * The algorithm for traversing the binary tree is based on that
 * given in
 * https://www.geeksforgeeks.org/binary-tree-array-implementation/
 *
 * Assume that nLayers is the number of layers of conditions
 * (3 for this problem)
 * Array conditions of dimension 2^nLayers - 1 stores name of
 * conditions
 * Array diagnoses of dimension 2^(nLayers+1) - 1 stores name of
 * diagnoses
 * Array symptoms of dimenson 2^nLayers - 1 stores inquiry result
 * about corresponding conditions
 *
 * The pseudo code is given below
 *
 * COVID 19 Diagnosis: Main Algorithm
 * 1. Initialize array conditions
 * 2. Initialize array diagnoses
 * 3. Initialize array symptoms
 * 4. inquiry to get symptoms
 * 5. make diagnosis
 * 6. output diagnosis
 *
 * inquiry to get symptoms
 * 1. index = 0
 * 2. nLayers = 3
 * 3. for iLayer = 0 to nLayers-1
 *    {
 * 3.1   if conditions[index] is NULL, break
 * 3.2   write a question based on conditions[index]
 * 3.3   read symptom to symptoms[index]
 * 3.4   index = NewIndex(index, symptoms)
 *     }
 *
 * make diagnosis
 * 1. index = 0
 * 2. for iLayer = 0 to nLayers-1
 *    {
 * 2.1   index = NewIndex(index, symptoms)
 * 2.2   if index >= 2^nLayers - 1 or
 *      symptoms[index] is NULL,
 * 2.2.1    break
 *     }
 * 3. diagnosis = diagnoses[index]
```

```
 *
 * NewIndex(index, symptoms)
 * 1. if symptoms[index] is YES
 *    {
 * 1.1     result = 2 * index + 1
 *     }
 * 2. else if symptoms[index] is NO
 *    {
 *        result = 2 * index + 2
 *     }
 * 3.  else
 *    {
 *         should not be here, throw an exception
 *     }
 * 4.  return result
 */

using System;

namespace Problem7
{
  enum Symptom
  {
    NULL = 0,
    YES = 1,
    NO = -1
  }

  class Program
  {
    static int NewIndex(int index, Symptom[] symptoms)
    {
      int result = 0;
      Symptom symp = symptoms[index];
      if(symp == Symptom.YES)
      {
        result = 2 * index + 1;
      }
      else if(symp == Symptom.NO)
      {
        result = 2 * index + 2;
      }
      else
      {
        throw new Exception("Unexpected symptom");
      }
      return result;
    }

    static void InitializeSymptoms(out Symptom[] symptoms)
    {
      symptoms = new Symptom[] {
        Symptom.NULL,
        Symptom.NULL,
        Symptom.NULL,
        Symptom.NULL,
        Symptom.NULL,
        Symptom.NULL,
        Symptom.NULL,
    };
    return;
```

```
}

static void Inquiry(int N_LAYERS, string[] conditions,
  ref Symptom[] symptoms)
{
  string condition = "";
  string answer = "";
  Symptom symptom = Symptom.NULL;
  int index = 0;
  for(int iLayer = 0; iLayer < N_LAYERS; ++iLayer)
  {
    condition = conditions[index];
    if(condition == "") break;
    Console.Write("是否符合" + condition + "? (Y/N)  ");
    answer = Console.ReadLine();
    if(answer.Substring(0, 1) == "Y" ||
       answer.Substring(0, 1) == "y")
    {
      symptom = Symptom.YES;
    }
    else
    {
      symptom = Symptom.NO;
    }
      symptoms[index] = symptom;
      index = NewIndex(index, symptoms);
  }
  return;
}

static string MakeDiagnosis(int N_LAYERS, Symptom[] symptoms,
  string[] diagnoses)
{
  string diagnosis = "";
  int index = 0;
  for(int iLayer = 0; iLayer < N_LAYERS; ++iLayer)
  {
    index = NewIndex(index, symptoms);
    if( (index >= (int) Math.Pow(2, N_LAYERS) - 1) ||
        (symptoms[index] == Symptom.NULL) )
      break;
  }
  diagnosis = diagnoses[index];
  return diagnosis;
}

static void Output(string diagnosis)
{
  Console.WriteLine("Diagnosis for COVID 19: " + diagnosis);
}

static void Main(string[] args)
{
  const int N_LAYERS = 3;
  string[] conditions;
  InitializeConditions(out conditions);
  string[] diagnoses;
  InitializeDiagnoses(out diagnoses);
  Symptom[] symptoms;
  InitializeSymptoms(out symptoms);
  Inquiry(N_LAYERS, conditions, ref symptoms);
```

```
    string diagnosis = MakeDiagnosis(N_LAYERS, symptoms, diagnoses);
    Output(diagnosis);

    Console.WriteLine("\n 按 enter 或 retuen 鍵結束");
    Console.ReadLine();
  }
 }
}
```