通識計算機程式設計期中考參考解答

臺灣大學 鄭士康

(答案達到要求即可，不需要和這裡的解答一模一樣)

滿分 100

本講義除另有註明外，採創用CC姓名標示-非商業性-相同方式分享3.0臺灣版授權釋出

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) (3%)
```
bool valid;
double xd;
float pr;
```

(b) (3%)
```
Console.Write("輸入一個帶有小數點的實數: ");
```

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

(d) (3%)
```
pr = (float)xd;
```

(e) (3%)
```
valid = (pr >= 0.0f && pr <= 1.0f);
```

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) (3%)
```
++n;
```

(b) (3%)
```
gene = (int)(chromosome * 2);
```

(c) (3%)
```
double result = 1.0 - Math.Abs(x);
```

(d) (3%)
```
bit = (bit > 0.0) ? 0 : 1;
```

(e) (3%)
```
char c = '\t';
```

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) (3%)
```
Random rand = new Random();
double chromosome = rand.NextDouble();
```

(b) (3%)
```
const int POPULATION_SIZE = 100;
```

(c) (3%)
```
for(int i = 0; i < POPULATION_SIZE; ++i)
{
  fits[i] = double.Parse(Console.ReadLine());
}
```

(d) (3%)
```
Array.Sort(fits);
Array.Reverse(fits);
double optimumValue = fits[0];
```

(e) (3%)

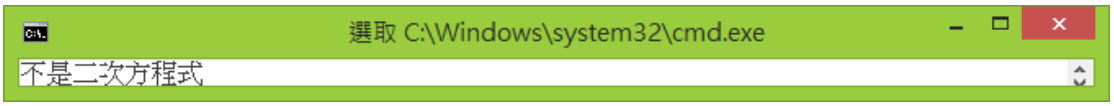
```
static void Initialize(Random rand,
    Out double chromosome)

{
    chromosome = rand.NextDouble();
}
```

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) (6%) (兩個語法錯誤)

```
int i = 0;
bool iEqualsZero = (i == 0); // 「相等」的符號是「==」,
                             // 不是代表assignment的
                             // 「=」
Console.WriteLine("\"i equals zero\" is " +
    iEqualsZero);  // 作為字串內容一部分的雙引號前，須加上
                   // escape 符號 「\」
```

(b) (3%) (一個語義錯誤) 執行時螢幕應顯示



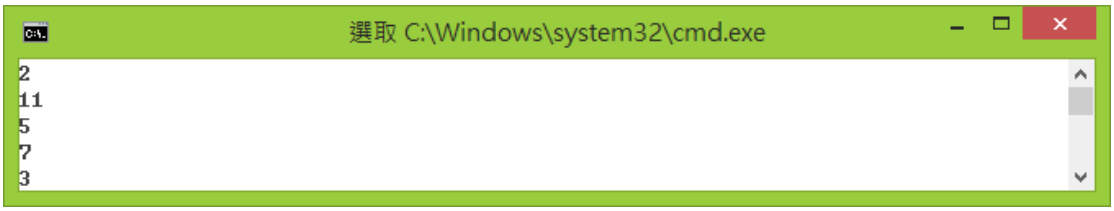

```
int a = 0;
double delta = 4.0;
if (a != 0)
    if (delta > 0)
        Console.WriteLine("兩實根");
else  // 懸置if，對應的是最近的if敘述「if(delta > 0)」
      // 所以「Console.WriteLine("不是二次方程式");」
      // 會併入if(delta > 0)敘述的else部分，不可能被執行
      // (因為 a != 0 不成立)
    Console.WriteLine("不是二次方程式");
```

應用大括弧確定**if…else…**影響範圍，更正為

```
int a = 0;
```

```
double delta = 4.0;
if (a != 0)
{
    if (delta > 0)
    {
        Console.WriteLine("兩實根");
    }
}
else
{
    Console.WriteLine("不是二次方程式");
}
```

(c) (3%) (一個語義錯誤) 執行時螢幕應顯示




```
int[] p = { 2, 11, 5, 7, 3 };
int[] q = p;  // 淺層複製, reference p 和 q 指向相同的
              // heap memory 區域,
              // 導致q內容的變動等於p的內容跟著改變
              // 無法輸出原先p 陣列內容
Array.Sort(q);
for(int i = 0; i < p.Length; ++i)
{
    Console.WriteLine(p[i]);
}
```

改用 deep copy, 更正為

```
int[] p = { 2, 11, 5, 7, 3 };
int[] q = new int[p.Length]; // reference q 指向自己的
                             // 新 heap memory 區域,
                             // 與 p 無關
Array.Copy(p, q, p.Length);  // 深層複製, reference p
                             // 和 q 各指向原先
                             // heap memory 區域,
```

```
                                        // 只是內容複製而已
                                        // 導致q內容的變動與p無關
Array.Sort(q);
for(int i = 0; i < p.Length; ++i)
{
    Console.WriteLine(p[i]);        // 輸出原先p 陣列內容
}
```

(d) (3%) 數個同類型語法錯誤

```
static void Main(string[] args)
{
   double x = 1.0;
   double y = -1.0;
   double u;  // 須設定初值才能作為傳址呼叫的參數
   double v;  // 須設定初值才能作為傳址呼叫的參數
   Transform(x, y, ref u, ref v);
}

static void Transform(double x, double y,
    ref double u, ref double v)
{
    double a11 = 0.3;
    double a12 = 0.5;
    double a21 = 0.7;
    double a22 = 0.1;
    u = a11 * x + a12 * y;
    v = a21 * x + a22 * y;
}
```

設定**u**, **v**初值即可. 但是使用out參數更合原程式碼用意. 更正為

```
static void Main(string[] args)
{
   double x = 1.0;
   double y = -1.0;
   double u;
```

```
        double v;
        Transform(x, y, out u, out v);
    }

    static void Transform(double x, double y,
         out double u, out double v)
    {
         double a11 = 0.3;
         double a12 = 0.5;
         double a21 = 0.7;
         double a22 = 0.1;
         u = a11 * x + a12 * y;
         v = a21 * x + a22 * y;
    }
```

5. 試寫出下列程式的螢幕輸出 (5 %)

```
選取 C:\WINDOWS\system32\cmd.exe
x = 0.5
Encoded as:
genes[0] = 1
genes[1] = 0
genes[2] = 0


genes are:
genes[0] = 0
genes[1] = 1
genes[2] = 1

Decoded as: 0.375
```

6. 試寫出下列程式的螢幕輸出 (10 %)

```
選取 C:\WINDOWS\system32\cmd.exe
Before roulette wheel selection
fits[0] = 28, chromosomes[0] = 1
fits[1] = 18, chromosomes[1] = 2
fits[2] = 14, chromosomes[2] = 3
fits[3] = 9, chromosomes[3] = 4
fits[4] = 26, chromosomes[4] = 5
After roulette wheel selection
fits[0] = 26, chromosomes[0] = 5
fits[1] = 28, chromosomes[1] = 1
fits[2] = 28, chromosomes[2] = 1
fits[3] = 26, chromosomes[3] = 5
```

7. 基因演算法(Genetic Algorithm, 簡稱 GA)

```
/*
 * Simple Genetic Algorithm
 *
 * 參考
http://people.sc.fsu.edu/~jburkardt/cpp_src/simple_ga/simple_ga.cpp
 *
 * 虛擬碼
 *
 * 1.    產生第一代的chromosome族群，並計算每個chromosome的適配值
 * 2.    Repeat N_GENERATIONS 個世代
 *      {
 * 2.1       以俄羅斯輪盤選出參加交配的親代parent chromosomes
 * 2.2       Repeat POPULATION_SIZE/2 次
 *          {
 * 2.2.1     由parent chromosomes隨機挑選兩個chromosome
 * 2.2.2     以單點交叉交換基因, 產生兩個新的offspring chromosome
 *          }
 * 2.3       依照突變機率模擬每個offspring chromosome內的基因突變
 * 2.4       以offspring chromosomes 取代原先的chromosomes族群
 * 2.5       計算新的chromosomes族群中每個chromosome的適配值
 *      }
 * 3.    以族群各chromosome的適配值為key，對應染色體為item，依照
 *        key由大而小排序
 * 4.    輸出最大適配值(optimum value)及對應染色體(optimum
 *        solution)
 *
```

```
 *
 * Behavior 測試規劃
 *
 * population size: 100 (chromosomes)
 * genes in a chromosome: genes[0], ..., genes[31]
 * mutation probability: 0.15
 * fitness measure = 1.0 - Abs(x - 0.5)
 * theoretical best gene: x = 0.5
 * maximum number of generations: 1000
 *
 */

using System;

namespace Problem7
{
    class Program
    {
        static void Main(string[] args)
        {
            Random randForInitialization = new Random();
            Random randForSelection = new Random();
            Random randForChoosingMates = new Random();
            Random randForCrossover = new Random();
            Random randForMutation = new Random();
            double P_M = 0.15;
            const int POPULATION_SIZE = 100;
                                                // should be an even number

            double[] parents = new double[POPULATION_SIZE];
            double[] fits = new double[POPULATION_SIZE];
            double[] offsprings = new double[POPULATION_SIZE];
            int p1 = 0;
            int p2 = 0;
            const int N_GENES = 32;
            const int N_GENERATIONS = 1000;
            Initialize(randForInitialization,
                POPULATION_SIZE, out parents, out fits);
```

```
    for (int n = 0; n < N_GENERATIONS; ++n)
    {
        RouletteWheelSelection(fits, parents,
            randForSelection);
        for (int m = 0; m < POPULATION_SIZE / 2;
            ++m)
        {
            p1 = randForChoosingMates.Next() % POPULATION_SIZE;
            p2 = randForChoosingMates.Next() % POPULATION_SIZE;
            Crossover(parents[p1], parents[p2], N_GENES,
                randForCrossover,    out offsprings[2 * m],
                out offsprings[2 * m + 1]);
        }
        for (int q = 0; q < POPULATION_SIZE; ++q)
        {
            Mutate(offsprings[q], N_GENES, P_M,
                 randForMutation);
        }
        Replace(parents, offsprings);
        fits = Evaluate(parents);
    }
    Array.Sort(fits, parents);
    Array.Reverse(fits);
    Array.Reverse(parents);
    Console.WriteLine("optimal x = {0} ", parents[0]);
    Console.WriteLine("optimal value = {0} ", fits[0]);
}

public static void Initialize(Random rand, int populationSize,
    out double[] chromosomes, out double[] fits)
{
    double x;
    chromosomes = new double[populationSize];
    fits = new double[populationSize];
    for (int i = 0; i < populationSize; ++i)
    {
        x = rand.NextDouble();
        chromosomes[i] = x;
```

```
        fits[i] = Fitness(x);
    }
}

// One-Point crossover
public static void Crossover(double parent1, double parent2,
     int nGenes, Random rand,
     out double offspring1, out double offspring2)
{
    int[] genes1 = Encode(parent1, nGenes);
    int[] genes2 = Encode(parent2, nGenes);
    int k = rand.Next() % (nGenes + 1);
    int temp;
    for (int i = k; i < nGenes; ++i)
    {
        temp = genes1[i];
        genes1[i] = genes2[i];
        genes2[i] = temp;
    }
    offspring1 = Decode(genes1);
    offspring2 = Decode(genes2);
}

// Mutate
public static double Mutate(double chromosome, int nGenes,
     double pm, Random rand)
{
    int[] genes = Encode(chromosome, nGenes);
    double r = 0.0;
    for (int i = 0; i < nGenes; ++i)
    {
        r = rand.NextDouble();
        if (r < pm) genes[i] = (genes[i] == 1) ? 0 : 1;
    }
    double result = Decode(genes);
    return result;
}
```

```
    // Replace
    public static void Replace(double[] parents,
        double[] offsprings)
    {
        Array.Copy(offsprings, parents, offsprings.Length);
    }
}
```