通識計算機程式設計期末考

1/13/2017

試題共 7 題，兩面印製 27 頁，滿分 103

1. 本題假定要畫出如圖 1 所示台大校歌五線譜第一小節的五個音符。為簡化問題，以主控台輸出字串，代替真正畫圖的視窗圖形介面程式。依照所畫音符外貌不同，可以把要畫的音符分成四分之一音符、附點八分之一音符、十六分之一音符等，當然各音符都必須攜帶本身的音高(pitch)資訊，因此可以設定一個共通的抽象父類別，其中保管處理音高訊息。參考下頁 UML 類別圖(圖 2)，撰寫 C#敘述達成下列要求:(假設 **using System;** 敘述已經包含於程式中)。令音高的列舉型別定義如下：

```
enum Pitch
{
    C,              // Do
    C_SHARP,
    D,              // Re
    D_SHARP,
    E,              // Mi
    F,              // Fa
    F_SHARP,
    G,              // Sol
    G_SHARP,
    A,              // La
    A_SHARP,
    B               // Si
}
```

(a) 撰寫抽象類別 **MusicNote**，包含成員變數 **pitch** 及抽象函式 Draw。其中的列舉型別 **Pitch** 定義如上 。 (3%)

(b) 撰寫 **MusicNote** 建構式，設定其成員變數 **pitch** 之值。 (6%)

(c) 撰寫 **MusicNote** 中的成員函式 **PitchName**，將成員變數 **pitch** 由列舉型別 **Pitch** 轉換為字串，例如 **Pitch.C_SHARP** 轉為**"C#"**輸出。提示：利用 **switch** 敘述。(6%)

(d) 建立類別 **QuarterNote(四分之一音符)**，繼承 **MusicNote**，實作其建

構式及 **public 成員函式 Draw()**。提示：**Draw** 函式內的要寫出的文句可以利用繼承來的 **PitchName** 函式，取得音高 **pitch** 的對應字串：

```
"QuarterNote: " + PitchName()
```

(6%)

(e) 寫一段主程式類別 **Program** 中的 **static** 函式 **DrawAllNotes**，以 **MusicNote** 的陣列為輸入參數，呼叫每個陣列元素的 **Draw** 函式。(3%)

(f) 假定另外兩種音符 **DottedEighthNote**、**SixteenthNote** 也仿照 **QuarterNote** 完成了。寫一段測試主控台主程式，建立五個元素的 **MusicNote** 陣列 **notes**，令 **notes[0]** 到 **notes[4]** 分別為 **QuarterNote**、**DottedEighthNote**、**SixteenthNote**、**QuarterNote**、**QuarterNote** 物件；對應的音高依固定唱名法，分別為 **Pitch.F**、**Pitch.D**、**Pitch.D_SHARP**、**Pitch.F**、**Pitch.A_SHARP**。接著呼叫函式 **DrawAllNotes**，輸出五個音符的種類及對應音高如圖 3。(6%)

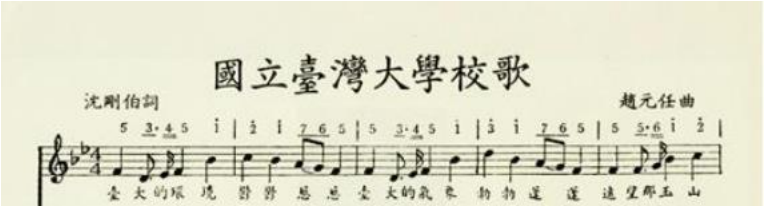


圖 1. 台大校歌的起始部分。原件由台大圖書館特藏組保存。

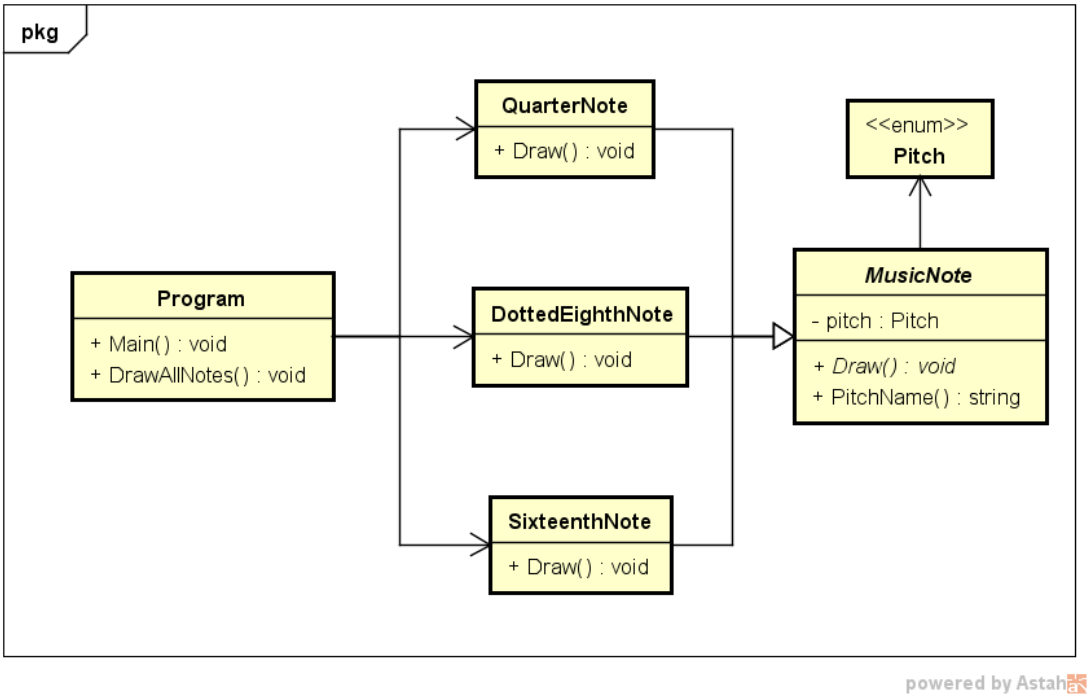


圖 2. 第 1 題對應的 UML 類別圖

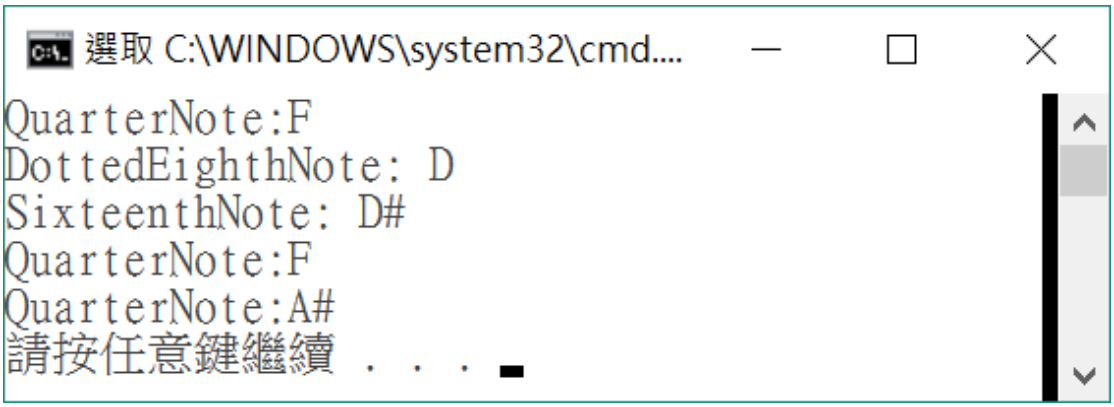


圖 3. 第 1(f)題程式碼執行主控台螢幕畫面

2. 找出以下程式片段之錯誤，並予更正.

(a) (3%) 一個錯誤

```
class TestA_Parent
{
    private int m;
    public TestA_Parent()
    {
        m = 3;
    }
    public int M
    {
        get { return m; }
    }
}

class TestA : TestA_Parent
{
    private int n;
    public TestA()
        : base()
    {
        n = 2;
    }
    public int sum()
    {
```

```
        return m + n;
    }
}
```

(b) (3%) 一個錯誤

```
class TestB
{
    private int m;
    public TestB(int m)
    {
        this.m = m;
    }
    public static int FuncB()
    {
        return m;
    }
}
```

(c) (3%) 一種錯誤。

```
class TestC
{
    private int m;
    public TestC(int m)
    {
        this.m = m;
    }

    public int M
    {
        get { return m; }
    }

    // 要把成員變數m除以2
    public void Func(int m)
    {
        m /= 2;
    }
}
```

(d) (3%) 一種錯誤

```
class TestD
{
    private int m;
    public TestD()
    {
        m = 0;
    }
    public int M
    {
        set { m = value; }
        get { return m; }
    }
    // 此處須加入複製建構式的宣告
}

class Program
{
    static void Main(string[] args)
    {
        TestD d1 = new TestD();
        TestD d2 = new TestD();

        // 錯誤! 這一行需改為利用複製建構式的寫法
        d2 = d1;

        // 要讓d1和d2的m分別等於15和40
        d1.M = 15;
        d2.M = 40;
    }
}
```

(e) (3%) 一種錯誤

```
interface TestE_IF
{
    int Triple();
    void SetM(int m);
}
```

```
    class TestE : TestE_IF
    {
        private int m;
        public TestE()
        {
            m = 5;
        }
        public int Triple()
        {
            return 3 * m;
        }
    }
```

3. 試寫出下列程式的輸出 (12%)

```
// Problem3.Program.cs
using System;
namespace Problem3
{
    class Program
    {
        static void Main(string[] args)
        {
            PluckedString pluckedString = new PluckedString(6, 1, 4);
            for(int t = 0; t < 10; ++t)
            {
                double y = pluckedString.NextSample(2);
                Console.WriteLine("t = {0}, y = {1}", t, y);
                pluckedString.Update();
            }
        }
    }
}

//Problem3.PluckedString.cs
using System;
namespace Problem3
{
```

```
public class PluckedString
{
    private int railLength;
    private ForwardLine upperRail;
    private BackwardLine lowerRail;

    public PluckedString(int railLength, int pick, int amplitude)
    {
        this.railLength = railLength;
        double[] initialShape = InitialShape(pick, amplitude);
        upperRail = new ForwardLine(railLength, initialShape, 0.5);
       lowerRail = new BackwardLine(railLength, initialShape, 0.5);
    }

    public void Update()
    {
        upperRail.Update();
        lowerRail.Update();
    }

    public double NextSample(int pickUpLoc)
    {
        double outSampU = upperRail.Access(pickUpLoc);
        double outSampL = lowerRail.Access(pickUpLoc);
        double outSamp = outSampU + outSampL;
        return outSamp;
    }

    private double[] InitialShape(int pick, int amplitude)
    {
        double upSlope = amplitude / (double) pick;
       double downSlope = amplitude /
          (double)(railLength - pick - 1);
        double[] initialShape = new double[railLength];
        for(int i = 0; i < pick; ++i)
        {
            initialShape[i] = upSlope * i;
        }
```

```
            for(int i = pick; i < railLength; ++i)
            {
                initialShape[i] = downSlope * (railLength - 1 - i);
            }
            return initialShape;
        }
    }
}

//Problem3.DelayLine
using System;

namespace Problem3
{
    public class DelayLine
    {
        protected int length;
        protected double[] data;
        protected int pointer;

        public DelayLine(int length, double[] initialShape,
         double factor)
        {
            this.length = length;
            data = new double[length];
            for(int i = 0; i < length; ++i)
            {
                data[i] = initialShape[i] * factor;
            }
        }

        public virtual void Update() {}

        public double Access(int pickUpLoc)
        {
            int outLoc = pointer + pickUpLoc;
            while (outLoc < 0) outLoc += length;
```

```
            while (outLoc > length-1) outLoc -= length;
            return data[outLoc];
        }
    }

    public class ForwardLine : DelayLine
    {
        public ForwardLine(int length, double[] initialShape,
         double factor) : base(length, initialShape, factor)
        {
            pointer = 0;
        }

        public override void Update()
        {
            pointer = ++pointer;
            pointer = (pointer > length) ? 0 : pointer;
        }
    }

    public class BackwardLine : DelayLine
    {
        public BackwardLine(int length, double[] initialShape,
         double factor) : base(length, initialShape, factor)
        {
            pointer = length;
        }

        public override void Update()
        {
            pointer = --pointer;
            pointer = (pointer < 0.0) ? length : pointer;
        }
    }
}
```

4．試寫出以下程式在下列狀況時的主控台螢幕輸出。

(a) (3%) 檔案 **tortoisebot.urdf** 尚未建立。

(b) (3%) 檔案 **tortoisebot.urdf** 已在正確位置，且內容為

```
<?xml version="1.0"?>
<robot name="tortoisebot">
  <link name="base_link">
  </link>
  <link name="front_caster">
  </link>
  <joint name="front_caster_joint" type="continuous">
  </joint>
</robot>
```

(c) (3%) 檔案 **tortoisebot.urdf** 已在正確位置，且內容為

```
<robot name="tortoisebot">
  <link name="base_link">
  </link>
  <link name="front_caster">
  </link>
  <joint name="front_caster_joint" type="continuous">
  </joint>
</robot>
```

(d) (3%) 檔案 **tortoisebot.urdf** 已在正確位置，且內容為

```
<?xml version="1.0"?>
<robot name="tortoisebot">
  <link name="base_link">
  </link>
  <link name="front_caster">
  <joint name="front_caster_joint" type="continuous">
  </joint>
</robot>
```

==========

```
// Problem4
using System;
using System.IO;
using System.Collections.Generic;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;

namespace Problem4
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                string fileName = "tortoisebot.urdf";
                SimpleRobot tortoisebot = new SimpleRobot(fileName);
                Console.WriteLine();

                BinaryFormatter formatter = new BinaryFormatter();
                FileStream output = new
                   FileStream("tortoisebot_urdf.rob",
                                  FileMode.Create, FileAccess.Write);
                formatter.Serialize(output, tortoisebot);
                output.Close();
                Console.WriteLine();

                FileStream input = new
                   FileStream("tortoisebot_urdf.rob",
                                 FileMode.Open, FileAccess.Read);
                Object obj = formatter.Deserialize(input);
                if (obj.GetType() == tortoisebot.GetType())
                {
                    SimpleRobot robot = (SimpleRobot)obj;
                    Console.WriteLine("Robot name : " + robot.NAME);
                    Console.WriteLine("Robot links: ");
                    for (int i = 0; i < robot.N_LINKS; i++)
                    {
```

```
                Console.WriteLine("\t" + robot.LINKS[i]);
            }
            Console.WriteLine("Robot joints: ");
            for (int i = 0; i < robot.N_JOINTS; i++)
            {
                Console.WriteLine("\t" + robot.JOINTS[i]);
            }
        }
        else
        {
            throw new SerializationException();
        }
    }
    catch (AbnormalParsingException e)
    {
        Console.WriteLine(e);
    }
    catch (FileNotFoundException)
    {
        Console.WriteLine("File not found");
    }
    catch (SerializationException)
    {
        Console.WriteLine(
            "Error in serializing/deserializing objects");
    }
    catch (IOException)
    {
        Console.WriteLine(
            "Can not open or close file");
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
  }
}
```

```
[Serializable]
class SimpleRobot
{
    private string xml_version;
    private string name;
    private int nLinks;
    private string[] links;
    private int nJoints;
    private string[] joints;
    public SimpleRobot(string fileName)
    {
        try
        {
            xml_version = XMLVersion(fileName);
            name = RobotName(fileName);
            ParseForElements(fileName, "<link", "</link>",
               out nLinks, out links );
            ParseForElements(fileName, "<joint", "</joint>",
               out nJoints, out joints);
        }
        catch (Exception e)
        {
            Console.WriteLine(
              "Throw an exception from constructor of SimpleRobot");
            throw e;
        }
    }

    public string XML_VERSION  { get { return xml_version; } }
    public string NAME { get { return name; } }
    public int N_LINKS { get { return nLinks; } }
    public string[] LINKS { get { return links; } }
    public int N_JOINTS { get { return nJoints; } }
    public string[] JOINTS { get { return joints; } }

    private string XMLVersion(string fileName)
    {
        string content = "";
```

```
int length = 0;
string line = "";
bool beginMarkFound = false;
bool endMarkFound = false;
int begin = -1;
int end = -1;
char[] delimiters = new char[] { ' ', '=', '\"' };
try
{
    StreamReader input = new StreamReader(fileName);
    Console.WriteLine("XMLVersion: Open file");
    try
    {
        line = input.ReadLine();

        begin = line.IndexOf("<?xml");
        beginMarkFound = (begin >= 0);
        if (!beginMarkFound)
        {
            throw new AbnormalParsingException(
                "Symbol \"<?\" not found");
        }
        int beginIndex = begin + 5;

        end = line.IndexOf("?>");
        endMarkFound = (end >= 0);
        if (!endMarkFound)
        {
            throw new AbnormalParsingException(
                "Symbol \"?>\" not found");
        }
        length = end - beginIndex;
        if (length < 0)
        {
            throw new AbnormalParsingException(
                "Abnormal xml header");
        }
        content = line.Substring(beginIndex, length);
```

```
                string[] terms = content.Split(delimiters);
                return terms[3];
            }
            catch (AbnormalParsingException e)
            {
                Console.WriteLine(
             "Throw an abnormal-parsing exception from XMLVersion");
                throw e;
            }
            catch (Exception e)
            {
                Console.WriteLine(
          "Throw an exception from inner try-catch in XMLVersion");
                throw e;
            }
            finally
            {
                Console.WriteLine(
                "Enter inner finally in function XMLVersion");
                input.Close();
                Console.WriteLine("Close file");
            }
        }
        catch (AbnormalParsingException e)
        {
            Console.WriteLine(
"Throw an abnormal-parsing exception from outer try-catch in XMLVersion");
            throw e;
        }
        catch (FileNotFoundException e)
        {
            Console.WriteLine(
  "Throw a file-not-found exception from outer try-catch in XMLVersion");
            throw e;
        }
        catch (Exception e)
        {
            Console.WriteLine(
```

```
      "Throw an exception from outer try-catch in XMLVersion");
        throw e;
    }
}

private string RobotName(string fileName)
{
    string content = "";
    int length = 0;
    string line = "";
    bool beginMarkFound = false;
    bool endAngularBracketFound = false;
    bool endMarkFound = false;
    bool robotNameFound = false;
    int begin = -1;
    int endAngularBracket = -1;
    int end = -1;
    char[] delimiters = new char[] { ' ', '=', '\"' };

    try
    {
        StreamReader input = new StreamReader(fileName);
        Console.WriteLine("RobotName: Open file");
        try
        {
            while (!input.EndOfStream)
            {
                line = input.ReadLine();
                if (!beginMarkFound)
                {
                    begin = line.IndexOf("<robot");
                    beginMarkFound = (begin >= 0);
                }
                if (!beginMarkFound) continue;
                if (!endAngularBracketFound)
                {
                    endAngularBracket = line.IndexOf('>');
                    endAngularBracketFound = (
```

```
            endAngularBracket >= 0);
        }
        if (endAngularBracketFound && !robotNameFound)
        {
            int beginIndex = begin + 6;
            length = endAngularBracket - beginIndex;
            if (length > 0)
            {
                content = line.Substring(
                 beginIndex, length);
               string[] terms = content.Split(delimiters);
                name = terms[3];
                robotNameFound = true;
            }
            else
            {
                throw new AbnormalParsingException(
                    "Abnormal robot name");
            }
        }
        else
        {
            if (!endAngularBracketFound)
            {
                throw new AbnormalParsingException(
              "\'>\' not in the same line of \"<robot\"");
            }
        }
        if (!endMarkFound)
        {
            end = line.IndexOf("</robot>");
            endMarkFound = (end >= 0);
        }
        if (endMarkFound)
        {
            break;
        }
    }
```

```
                if (beginMarkFound && !endMarkFound)
                    throw new AbnormalParsingException(
                        "Abnormal robot name");
            }
            catch (AbnormalParsingException e)
            {
                Console.WriteLine(
              "Throw an abnormal-parsing exception from RobotName");
                throw e;
            }
            catch (Exception e)
            {
                Console.WriteLine(
           "Throw an exception from inner try-catch in RobotName");
                throw e;
            }
            finally
            {
                Console.WriteLine(
                "Enter inner finally in RobotName");
                input.Close();
                Console.WriteLine("Close file");
            }
        }
        catch (AbnormalParsingException e)
        {
            Console.WriteLine(
   "Throw an abnormal-parsing exception from outer try-catch in RobotName");
            throw e;
        }
        catch (FileNotFoundException e)
        {
            Console.WriteLine(
"Throw a file-not-found exception from outer try-catch in RobotName");
            throw e;
        }
        catch (Exception e)
        {
```

```
        Console.WriteLine(
       "Throw an exception from outer try-catch in RobotName");
        throw e;
    }
    return name;
}

private void ParseForElements(string fileName,
   string beginMark, string endMark,
   out int nElements, out string[] elements)
{
    string content = "";
    int length = 0;
    string line = "";
    bool beginMarkFound = false;
    bool endAngularBracketFound = false;
    bool endMarkFound = false;
    bool elementNameFound = false;
    int begin = -1;
    int endAngularBracket = -1;
    int end = -1;
    char[] delimiters = new char[] { ' ', '=', '\"' };

    const int MAX_N_ELEMENTS = 10;
    elements = new string[MAX_N_ELEMENTS];
    nElements = 0;

    try
    {
        StreamReader input = new StreamReader(fileName);
        Console.WriteLine("ParseForElements: Open file");
        try
        {
            while (!input.EndOfStream)
            {
                line = input.ReadLine();
                if (!beginMarkFound)
                {
```

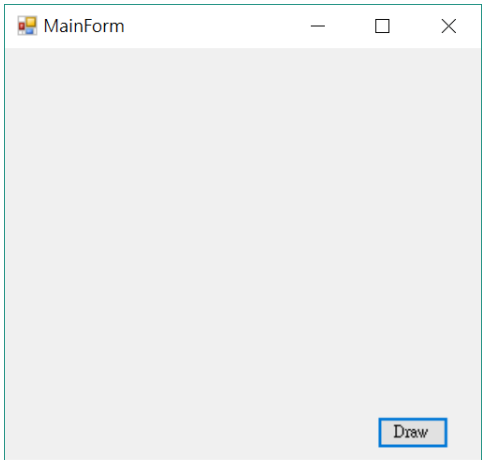
```
                begin = line.IndexOf(beginMark);
                beginMarkFound = (begin >= 0);
            }
            if (!beginMarkFound) continue;
            if (!endAngularBracketFound)
            {
                endAngularBracket = line.IndexOf('>');
           endAngularBracketFound = (endAngularBracket >= 0);
            }
            if (endAngularBracketFound && !elementNameFound)
            {
                int beginIndex = begin + beginMark.Length;
                length = endAngularBracket - beginIndex;
                if (length > 0)
                {
                 content = line.Substring(beginIndex, length);
                   string[] terms = content.Split(delimiters);
                    elements[nElements] = terms[3];
                    nElements++;
                    elementNameFound = true;
                }
                else
                {
                    throw new AbnormalParsingException(
"Abnormal " + beginMark.Substring(1, beginMark.Length - 2));
                }
            }
            else
            {
                if (!endAngularBracketFound)
                {
                    throw new AbnormalParsingException(
               "\'>\' not in the same line of " + beginMark);
                }
            }
            if (!endMarkFound)
            {
                end = line.IndexOf(endMark);
```

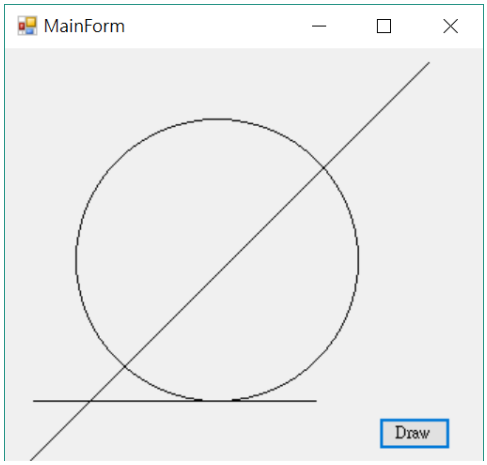
```
                endMarkFound = (end >= 0);
            }
            if (endMarkFound)
            {
                beginMarkFound = false;
                endAngularBracketFound = false;
                endMarkFound = false;
                elementNameFound = false;
                begin = -1;
                endAngularBracket = -1;
                end = -1;
            }
        }
        if (beginMarkFound && !endMarkFound)
            throw new AbnormalParsingException(
                endMark.Substring(1, endMark.Length - 1) +
                " not found");
    }
    catch (AbnormalParsingException e)
    {
        Console.WriteLine(
 "Throw an abnormal-parsing exception from ParseForElements");
        throw e;
    }
    catch (Exception e)
    {
        Console.WriteLine(
"Throw an exception from inner try-catch in ParseForElements");
        throw e;
    }
    finally
    {
        Console.WriteLine(
        "Enter inner finally in ParseForElements");
        input.Close();
        Console.WriteLine("Close file");
    }
}
```

```
            catch (AbnormalParsingException e)
            {
                Console.WriteLine(
"Throw an abnormal-parsing exception from outer try-catch in ParseForElements");
                throw e;
            }
            catch (FileNotFoundException e)
            {
                Console.WriteLine(
"Throw a file-not-found exception from outer try-catch in ParseForElements");
                throw e;
            }
            catch (Exception e)
            {
                Console.WriteLine(
              "Throw an exception from outer try-catch in ParseForElements");
                throw e;
            }
        }
    }

    class AbnormalParsingException : ApplicationException
    {
        private string message;
        public AbnormalParsingException(string message)
            : base()
        {
            this.message = message;
        }
        public override string ToString()
        {
            return message;
        }
    }
}
```

5.依據以下描述及程式框架，完成指定程式。你在答案卷只需寫下程式註解標示的部分。(6%)

程式描述：畫圓和直線。
建立如圖 4 之圖形使用介面，按下「**Draw**」按鈕(**button1**)，即於主視窗出現 1 **條斜線、1 條水平線、一個圓**，如圖 5 所示，均為黑色線條。斜線右上方端點座標為(300, 10)，左下方端點為(10, 300)；水平線左方端點座標為(20, 250)，長度為 200；圓形圓心為(150, 150)，半徑 100。





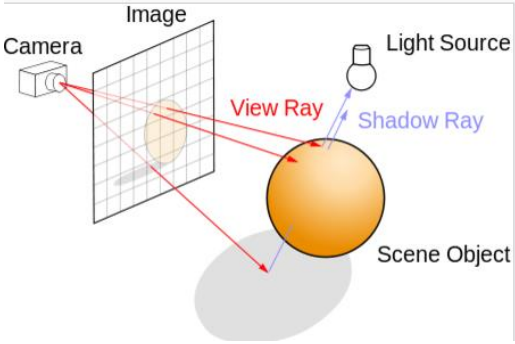
(左) 圖 4．程式開始執行時之視窗畫面
(右) 圖 5．按下「Draw」按鈕後截取之視窗畫面

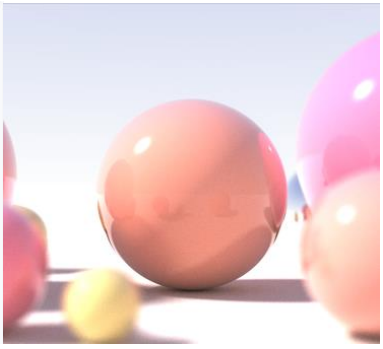
```
// Problem5.MainForm
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Problem5
{
    public partial class MainForm : Form
    {
```

```
        //***************************
        private bool started = false;
        //***************************

        public MainForm()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //*************************************
            started = true;
            DrawContents();
            //*************************************
        }

        //**********************************************
        private void DrawContents()
        {
              //*********************************************
              // 在此加入必要之程式碼
              //*********************************************
        }
        protected override void OnPaint(PaintEventArgs e)
        {
            base.OnPaint(e);
            if (started) DrawContents();
        }
        //**********************************************
    }
}
```

6. 3D 射線追蹤(ray tracing)繪圖是電腦遊戲、實況模擬、虛擬實境等的重要核心技術，其原理略如圖 6：由眼睛位置至假定的螢幕各像素(pixel)，產生許多射線。每條射線與虛擬環境中的物體可能產生許多交點(intersections)。這些交點中，與射線起始點(origin)最靠近的交點，稱為最近交點(nearest hit)。由最近交

點的物體顏色及多個光源的照明顏色及方位關係，可以決定出該條射線對應的螢幕像素應該呈現何種顏色，也可能發現交點對應到陰影，使其顏色較暗。如圖 7 就可能是這種演算法得到的螢幕影像。最近交點外的其他交點顯然被遮蔽，除非物體為可透視材質，否則不用考慮。另外，射線在最近交點處，可以應用光學原理，產生反射(reflection)射線及透射(refraction)射線，再追蹤下去，使螢幕上像素的顏色受多個物體顏色的影響：例如圖 7 中央圓球左下方部位的顏色，似乎看的到前方黃色小圓球的影響。執行射線追蹤時，通常會設定一些結束條件，例如反射或透射次數上限，或讓射線重要性逐漸降低到可忽略的大小。此外，射線追蹤方法，通常還會設定當射線沒有與任何物體相交時，讓對應螢幕像素顯現某種背景顏色，如圖 7 背景中的灰白色。由於 3D 射線追蹤大略符合自然界的光學原理，因此其畫面極為逼真。唯一的問題是需要大量計算，但因現代圖形顯示器(Graphic Processing Unit)效能極高，因此即使是畫面繁複的即時戰略遊戲，畫面變化依然十分流暢。





(左) 圖 6. 3D 射線追蹤繪圖原理示意
(右) 圖 7. 由圖 6 情境可能獲得的螢幕畫面示意
兩圖均取自 https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

射線追蹤時，最根本的計算是求取射線和情境中物體的最近交點。本題希望以簡化的二維問題，說明此一計算的概念。所假設的情境如圖 8 所示：射線 S 以(x_0, y_0)為起點(origin)，θ為入射角(射線箭頭反方向與正 x 軸所夾角)，向左下方射出。情境中的二維物體(可稱為 **Object2D**)包括一個圓 C 和一條水平線段 L。圖中射線與 C 和 L 有三個交點 P_1、P_2、P_3，顯然 P_1 是最近交點(nearest hit)。本題希望完成的程式，容易推廣至多個圓及水平線段的情境。

為求出射線 S 和 C、L 的所有交點，需要得出其數學式。射線 S 上的任一點(x, y)可以用參數式表為

$$x = x_0 - t\cos\theta$$
$$y = y_0 - t\sin\theta$$

其中$t > 0$代表由起點(x_0, y_0)沿箭頭方向量起，到(x, y)的距離。如果$t < 0$則(x, y)落在 S 反方向的射線上。

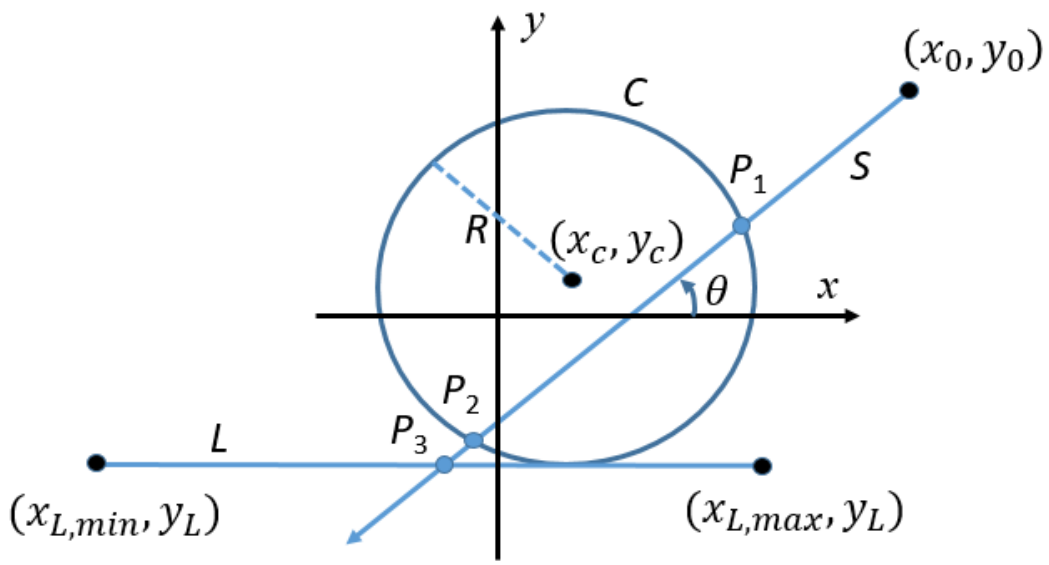


圖 8. 二維射線追蹤範例情境

圖 8 中的圓 C，圓心座標(x_c, y_c)，半徑 R，圓上點(x, y)的方程式為

$$(x - x_c)^2 + (y - y_c)^2 = R^2$$

交點 P_1、P_2同時落在 S 和 C 上，因此可以聯立兩者方程式，求出各自對應的參數

$$t = ((x_0 - x_c)\cos\theta + (y_0 - y_c)\sin\theta) \mp \sqrt{\Delta}$$

$$\Delta = R^2 - ((x_0 - x_c)\sin\theta - (y_0 - y_c)\cos\theta)^2$$

顯然判別式Δ為正、0、負時，參數 t 有 2、1、0 個解，對應兩個、一個、無交點。

水平線段 L 上的點(x, y)則滿足

$$x_{L,min} \le x \le x_{L,max}$$
$$y = y_L$$

交點 P_3由聯立 S 和 L 方程式，可算出參數 $t = \frac{y_0 - y_L}{\sin\theta}$，$x = x_0 - (y_0 - y_L)\cot\theta$。

當$x \ge x_{L,min}$且$x \le x_{L,max}$時，交點才存在，否則 S 不會打到 L。

求得射線 S 和 C、L 所有交點對應的 t 值後，以最小的正 t 值代入 S 參數式，就獲得最近交點的座標。

本題假設$(x_c, y_c) = (0,0)$，$R = 1$，$y_L = -1$，$x_{L,min} = -1.5$，$x_{L,max} = 0.5$。請撰寫一套程式，當使用者輸入射線起點座標x_0、y_0及θ角，輸出射線與所設圓及水平線段的最近交點。程式執行時的主控台畫面一例，如圖 9 所示。

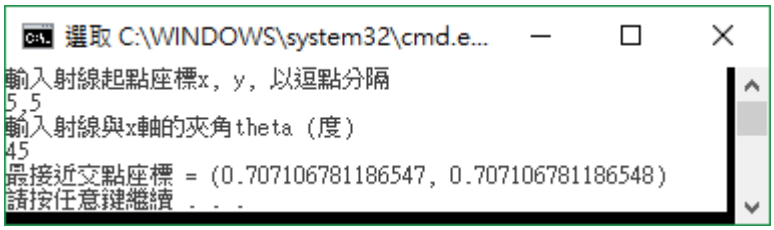


圖 9. 程式執行之主控台畫面一例

本題滿分 25 分，全部程式集中寫成一個大 **Main** 函式，不區分其他函式者，最高得 18 分；善用函式者，最高得 20 分；能利用虛擬碼或 UML 類別圖思考，適當劃分類別(class)者，最高得 22 分；善用類別多型(polymorphism)者，最高得 25 分。 (25%)

7. 請寫下本課程教學「待改進」之處及改進方法建議。 (3%)