1. return 的傳值、傳參考

第四個程式碼第六行後面的註解應該沒有「空字串」

```
#include <iostream>
#include <string>
using namespace std;

string foo();

int main() {
 string str = foo(); // 空字串

 cout << "address: " << &str
 << endl << str << endl;

 str = foo(); // 指定新字串，會複製一份

 cout << "address: " << &str
 << endl << str << endl;

 return 0;
}

string foo() {
 string s = "This is caterpillar speaking.";

 cout << "address: " << &s
 << endl << s << endl;

 return s;
}
```

2. 物件指標

第二個程式碼最後第三行

system("pause")

若在 linux 的環境下，應該要沒有這一行的

```
#include <iostream>
#include "SafeArray.h"
using namespace std;

int main() {
    SafeArray arrs[] = {SafeArray(5), SafeArray(10), SafeArray(15)};

    for(int i = 0; i < 3; i++) {
        SafeArray *safeArray = arrs + i;
        for(int j = 0; j < safeArray->length; j++) {
            safeArray->set(j, (j + 1) * 10);
        }
    }

    for(int i = 0; i < 3; i++) {
        SafeArray *safeArray = arrs + i;
        for(int j = 0; j < safeArray->length; j++) {
            cout << safeArray->get(j) << " ";
        }
        cout << endl;
    }

    system("pause");

    return 0;
}
```

3. 巢狀類別（Nested classes）

在第三和第四個程式碼之間的片斷程式碼

```
class PointDemo {
    ...
    friend class Point;   //  這一行應該要刪掉

private:
    // Nested Class

    class Point {
    …
    friend class PointDemo; //  這一行原本沒有，應該要加在 Point class 內啦~
    ...
    };
    ...
};
```

再下面的片斷程式碼

```
class PointDemo {
public:
    ...
    friend class Point;
private:
    // Nested Class
    class Point {
        ...
        friend class PointDemo; //  我覺得這一行要刪掉
        ...
    };
    ...
};
```

4. 使用 friend 函式重載運算子

最後四個小片段程式的 const 應該都要刪掉
因為有用到 p._x++;有更動到變數值

您也可以使用 friend 函式來重載++或--這類的一元運算子，但要注意的是，friend 不會有 this 指標，所以為了讓它具有++或--的遞增遞減 原意，您要使用傳參考的方式，將物件的位址告訴函式，例如：

```
class Point2D {
public:
    ....
    friend Point2D operator++(const Point2D&);  // 前置
    friend Point2D operator++(const Point2D&, int); // 後置

private:
    int _x;
    int _y;
};
```

實作時如下：

```
#include "Point2D.h"
....
Point2D operator++(const Point2D &p) {
    p._x++;
    p._y++;

    return p;
}

Point2D operator++(const Point2D &p, int) {
    Point2D tmp(p._x, p._y);

    p._x++;
    p._y++;

    return tmp;
}
```

5. 純虛擬函式、抽象類別 （Abstract class）

第二段第一行，一個類別中如果含有「純」虛擬函式…

一個類別中如果含有純虛擬函式，則該類別為一「抽象類別」（Abstract class），該類別只能被繼承，而不能用來直接生成實例，如果試圖使用一個抽象類別來生成實例，則會發生編譯錯誤。

6. 名稱空間（Namespace）

程式碼 main.cpp 第二行

#include "Point.h" 應該要改為#include "Point2D.h"

```
#include <iostream>
#include "Point.h"
using namespace std;

int main() {
    d2d::Point2D p1(10, 10);

    cout << "(x , y) : ("
         << p1.x() << ", "
         << p1.y() << ")"
         << endl;

    return 0;
}
```

7. 未格式化檔案 I/O

第二段第一行

"要處理檔案的輸出入，您必須先 include <fstream>標頭，如果要處理檔案輸出"

最後一個要改成"輸入"

8. 格式化檔案 I/O

第三個程式碼

在 cout << "無法輸出目的檔" << endl;

下面應該要加上 fin.close()，因為，現在要離開程式，而 fin 以經開啟檔案了

9. 二進位檔案 I/O

第二個程式碼

在 cout << "檔案讀入失敗" << endl;

下面應該要加上 fin.close()，因為，現在要離開程式，而 fin 以經開啟檔案了

10. get 指標與 put 指標

第一個程式碼第 12 行最後的分號應該要刪掉

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ifstream fin("data.txt");
    if(!fin) {
        cout << "無法讀取檔案\n";
        return 1;
    }

    char name[30];
    int request, account, score;

    cout << "輸入選項:" << endl;
         << "1) 顯示所有學生與分數" << endl
         << "2) 顯示及格學生與分數" << endl
         << "3) 顯示不及格學生與分數" << endl
         << "4) 離開" << endl;
```