《程式語言》

一、請問下列Java程式碼編譯後輸出結果為何？（20分）

```
Class tree{
    void message(){ System.out.println("1359"); }
    int tree(){ System.out.println("Tr"); return 3; }
}

class two extends tree{
    void message(){ super.message();
                    System.out.println("4321"); }
    two(){ System.out.println("Tw"); }
}

public class one extends two{
    void message(){ System.out.println("1234"); }
    one(){ System.out.println("On"); }
    public static void main (String[] args){
        one Exam = new one();
        Exam.message();
    }
}
```

試題評析	此題考物件導向中繼承及附載的觀念，值得注意的是此題含有陷阱，在tree中tree()方法因回傳型態為int並非建構函數，因此即使程度好的考生一樣很可能會因此而失去分數。
考點命中	《高點・高上程式語言講義》第四回，金乃傑編撰，頁37～38。

答：

程式輸出如下：
Tw
On
1234

說明：物件建構時會從祖父的建構子開始往自己呼叫，由於該程式中tree的tree()函數回傳型態為int，並非建構子，因此不會呼叫到，故先印出Tw；而message()函數因被one類別override，因此印出1234之結果。

二、請觀察以下程式碼：

(一)請問以下C++程式碼輸出答案為何？（5分）

(二)請問執行至return0時，是否回收原分配給物件p之動態記憶體？（5分）

(三)請說明此方法為解決何種問題？且程式概念為何？（10分）

```
class Smart{
    public:
    explicit Smart(int *p = NULL) { sp = p; }
    ~Smart() { delete(sp); }
```

```
    int& operator*() { return *sp; }
    private:
    int *sp;
};
int main(void){
    Smart p(new int());
    *p = 2016;
    cout << *p << endl;
    return 0;
}
```

試題評析	此題為C++的物件導向。在此題中必須要了解C++建立與解構物件的時機與方式，還需要有運算子超載的知識才能正確作答。若程度好的考生細心留意，應仍可獲得不錯的分數。
考點命中	1.《高點・高上程式語言講義》第三回，金乃傑編撰，頁30。 2.《高點・高上程式語言講義》第四回，金乃傑編撰，頁99～100, 127。

答：

(一)程式輸出如下：

2016

(二)會收回。因為物件 p 其實是在 main 領域({})中宣告而建立的區域變數，屬於堆疊動態變數，而非用 new 產生的外顯堆積變數，因此遇到函數 return 時會釋放活動紀錄，自動收回記憶體空間。

(三)解決記憶體洩漏（Memory Leakage）問題。此方法透過物件 p 的建構子、解構子管理物件中使用到的記憶體空間。在物件 p 初始化時傳入的整數物件會在物件 p 解構時隨著解構子被呼叫而釋放。如此可以避免程式設計師在撰寫程式時沒有手動釋放整數物件，使得 Heap 中殘留記憶體垃圾情形。

此程式主要概念是讓程式設計師要使用整數物件時先透過 Smart 管理，再使用 overloading 過的星號(*)函數讓程式設計師存取傳入的整數物件，使操作物件與操作一般的參考方法相同，可以輕易地存取數值。當離開副程式時，因為 p 物件是宣告在副程式區域變數中的物件，因此系統會自動呼叫解構子進行解構。而解構子中則透過 delete 釋放傳入的整數物件，消除記憶體垃圾。

說明：建構函數中explicit的用法是規定要使用「外顯」的方式初始化Smart物件。若不加上此關鍵字，則可使用如下語法建立物件：

Smart p = new int();

直接用等號指派而不是放在參數中傳入。

三、觀察以下C語言之程式，試問輸出為何？（20分）

```
int main(){
    int a = 1, b = 2, c = 3, d = 4;
    printf( "%d\n%d\n%d\n%d\n",  a+b+c+d, (b *= a),
            (a += d), (d++));
}
```

試題評析	此題若不是故意考驗考生是否了解C語言函數參數無特定執行順序的話，則大失國考水準！
考點命中	《高點・高上程式語言講義》第一回，金乃傑編撰，頁39及上課補充。

答：

程式輸出如下：

26

12

6
4

註：此題只有最後一個答案「4」可以確定，因為在標準的C語言（C99及C89）中，並未定義函數呼叫中參數執行的順序，又因printf中眾多參數都會修改到變數值（改到a及b），產生副作用，所以在不同編譯器下執行會有不同結果。以下列出C99官方文件之原文：
§6.5.2.2p10:
The order of evaluation of the function designator, the actual arguments, and subexpressions within the actual arguments is **unspecified**, but there is a sequence point before the actual call.

但實務上大多數編譯器會由右往左進行參數解析，因此最有可能跑出如上解答之答案。

四、請回答以下問題：
(一)請描述C#程式語言中關鍵字const之功用。（10分）
(二)請比較C#程式語言中關鍵字const與readonly差異。（10分）

試題評析	此題為C#題目，考到const及readonly關鍵字，若考生平日無充分C#開發經驗，則較難寫出具體的比較結果。
考點命中	《高點・高上程式語言講義》第三回，金乃傑編撰，頁75。

答：
(一)在 C#中 const 關鍵字可宣告常數屬性或區域常數。常數屬性為類別中的屬性，在 C#中又稱為欄位（field）；區域常數是宣告在方法（函數）中的常數常。const 常數的內容可以是數值、布林值、字串或 null 的參考，與變數最大不同是其值無法更改。
如：
public const int a = 5;
如此 a 之數值即無法更改，為常數 5。
(二)將 C#中 const 與 readonly 之差異比較如下表：

	const	readonly
宣告方法	public const int a = 5;	public readonly int a = 5;
宣告處	類別中、方法中皆可宣告	只能在類別中，成為欄位（field）
初始化	宣告時必須直接初始化。	可在宣告時初始化，亦可透過建構函數初始化，因此每一個物件可以有不同的 readonly 值。
繫結時期	編譯時（靜態）	執行時（動態）
儲存位置	Code Segment	Heap（Data Segment）
被外部程式引用時的處理	引用者會將 const 編譯進程式碼中，因此若修改其值，引用者中的 const 數值不受影響。若要更新引用者中數值，則引用者亦必須重新編譯	引用者使用參考方始取得數值，因此修改後即可馬上反映到引用者程式中
執行效能	較高	較低
使用彈性	非常低	較高

五、請問下列C語言程式碼執行後輸出結果為何？（20分）

```
int main() {
    double d[10][20][30][40];
```

```
    printf("%d\n", (int) sizeof(d[1][5]));
    printf("%d\n", (int) sizeof(d[1][5][20]));
    printf("%d\n", (int) (d - &d[3]));
    printf("%d\n", (int) (d[3][4] - d[5][8]));
}
```

試題評析	此題為記憶體位置題，相似於98年高考，求陣列的記憶體位置及運算。若考生能掌握好記憶體取址符號之意義，實不難作答，若能謹慎答題，滿分應大有人在。
考點命中	《高點・高上程式語言講義》第三回，金乃傑編撰，頁11～13。

答：

程式輸出結果如下：

(一)9600

d[1][5]代表一二維陣列，其維度為 30 * 40，陣列每個元素為倍精數，故計算所占空間為 30 * 40 * 8 = 9600

(二)320

d[1][5][20]代表一一維陣列，其大小為 40 個元素，考慮元素為倍精數，得所占空間為 40 * 8 = 320

(三)-3

可把 d 視為&d[0]，因此原算是等於&d[0] - &d[3]，此寫法將 d 視為一個極大的一維陣列，因此計算 d[0]到 d[3]的相差 3 格，則 d[0] - d[3]則為-3。

(四)-1320

可先把 d[3][4]與 d[5][8]視為一大型二維陣列 d[10][20]的兩個元素，則使用陣列公式計算其偏移量為 44；又 d[3][4]其實為&d[3][4][0]；d[5][8]其實為&d[5][8][0]，又第三個維度每個元素有 30 格，故得兩數值相減-44 * 30 = 1320。