《程式語言》

試題評析	第一題：本題考點爲 C++的建構與解構函數，較常出現在選擇題中，屬於三等考試中較冷門的考點。必須要掌握在程式不同地方宣告的物件其建構、解構的順序才能完整拿分。 第二題：本題爲強弱型態，亦屬較少出現的考題，但因考點單純，只要能分辨強、弱型態及隱性轉換，應不難拿分。 第三題：本題爲簡單的遞迴與活動紀錄的考題，若能知道活動紀錄欄位的意義，分數應容易好掌握。 第四題：本題結合垃圾回收與 Java 的解構機制，並非新考題，亦屬於基本分。 第五題：本題將網頁應用程式分類爲伺服端與用戶端，讓考生進行分類，屬新增考題，但大部分語言同學都已熟悉，程度中等同學仍應可拿取部分分數。
考點命中	第一題：《高點程式語言講義》第四回，金乃傑編撰，頁 79-83。 第二題：《高點程式語言講義》第三回，金乃傑編撰，頁 58-60。 第三題：《高點程式語言講義》第二回，金乃傑編撰，頁 58-65。 第四題：《高點程式語言講義》第三回，金乃傑編撰，頁 29-32。 第五題：《高點程式語言講義》第五回，金乃傑編撰，頁 28-65 及上課補充。

一、請回答下列有關 C++程式的問題。（每小題 10 分，共 20 分）

(一)如果執行起來，請問執行的螢幕輸出為何？

(二)請解釋每一行輸出是由那個物件在那個時間點前後（請參考程式所標示的時間點 1,2,3,4,5）印出的？

```
#include “car.h”
#include <iostream>
using namespace std;
Car::Car(){     cout<<   “Car is constructed”<<endl;}
Car::~Car(){     cout<<   “Car is destroyed”<<endl;}
//-------------------------------------------------
Car honda;
1:main() {
2:Car bmw;
3:Car *camery = new Car();
4:        honda.color =1;
5:        camery.color = 100;
6:}
```

【擬答】

螢幕輸出與時間點如下表所示：

輸出內容	時間點	說明
Car is constructed	1 之前	全域變數 honda 建構時呼叫建構子所印出
Car is constructed	2 執行時	main 區域變數 bmw 建構時呼叫建構子所印出
Car is constructed	3 執行時	main 區域變數指標 camery 指向之物件被建構時呼叫建構子所印出
Car is destroyed!	6 之前	主程式結束時刪除 bmw 物件
Car is destroyed!	6 之後	整個程式結束時刪除 honda 物件

說明：

honda 爲全域變數，在程式載入時便會建立，建構子即會被呼叫到；而整個程式鑰結束時亦會自動呼叫解構子。

bmw 爲區域變數，在 main 中宣告時建立，而在 main 結束時會呼叫解構子釋放。

camery 爲指標，指向一個 new 出來的物件，在 new 的時候呼叫建構子。而 new 要搭配 delete 手動釋放，不然物

件會儲存在 heap 裡面，帶程式結束後被作業系統收回，故不會印出解構子內容。此外，此程式第 5 行應該改爲 camery->color，不然程式無法執行。

二、程式語言的設計通常會決定採取 strong typing 或 weak typing 的設計。

(一)請運用你所熟知的程式語言舉一例 strong typing。(6 分)

(二)請運用你所熟知的程式語言舉一例 weak typing。(6 分)

(三)有些語言有所謂的 implicit type conversion，請舉例。(8 分)

【擬答】

(一)強型態(Strong Typing)：若一程式語言爲強型態語言，則該語言變數只有一個固定的資料型態，若變數儲存不同資料型態之值，其值的型態可以靜態或動態的檢查出來、變數在執行程式的過程中不能更改其資料型態及不允許編譯器對變數做隱性型態轉換。舉例而言 Ada 爲一強型態語言，考慮以下程式碼：

```
X : Integer:= 4;     --將 X 變數設爲整數的 4
Y : Float;           --將 Y 變數設爲浮點數
Y := X;              --錯誤！整數不能直接指派給浮點數
Y := 3.14;           --將 Y 設爲 3.14
X := Y               --錯誤！浮點數不能直接指派給整數
```

若要達到指派，則必須手動進行型態轉換，如下：

```
Y := float(X);       --手動將整數 X 轉爲浮點數才能指派給 Y
X := Integer(Y);     --手動將浮點數 Y 轉爲整數才能指派給 X
```

(二)弱型態(weak typing)：若一程式語言爲弱型態語言，則該語言變數的資料型態可以靜態也可以動態決定、變數在執行程式的過程中可以更改其資料型態、運算式中不必檢查運算元的資料型態相容性、允許編譯器對變數做隱性型態轉換。舉例而言，JavaScript 爲一弱型態語言，考慮以下程式碼：

```
var display = 5;              //將 display 變數設爲整數的 5
display = “give me ”+5;  //將 display 設定爲字串“give me ”加上 5，此時 display 被轉型爲一個字串
document.write(display);//印出結果“give me 5”
```

在弱型態語言中，變數的型態可以在執行中自由轉換，且轉型時也不需要特別的宣告，直譯(編譯)器便能自動將型態轉換。

(三)implicit type conversion 即爲「隱性轉換」，意思是變數的型態轉換不需要經過特定的符號即可自動完成，在分類上又可分爲「縮窄法」與「拓寬法」。考慮以下 C 語言程式：

```
int w = 3, x = 5;
double y = 5.5, z = 3.3;

w = y;  //將雙倍精度浮點數 y 指派給整數 w
        //在此指派中沒有任何的轉型符號，
        //而是由編譯器根據內定規則將 8 bytes IEEE 754 格式的 y
        //轉換爲 4 bytes 的整數 w
        //此轉換即爲隱性轉換，在轉換過程中因爲儲存的 byte 位數變少，
        //因此爲「縮窄法」。一般而言縮窄法會造成資料的遺失。

z = x;  //將整數 x 指派給雙倍精度浮點數 z
        //在此指派中沒有任何的轉型符號，
        //而是由編譯器根據內定規則將 4 bytes 的整數 x
        //轉換爲 8 bytes IEEE 754 格式的 z
        //此轉換即爲隱性轉換，在轉換過程中因爲儲存的 byte 位數增加，
        //因此爲「拓寬法」。一般而言拓寬法不會造成資料的遺失，
        //而且能表達的資料可以更精確。
```


三、下列是一個簡單的 C 遞迴副程式（recursive function）。請用這個例子來解釋編譯器在處理遞迴的

呼叫時，如何在那一種記憶體區塊處理與配置程式中的變數（a, c, d）。請繪製當 foo(1)被呼叫之後，並且遞迴到程式結束之間的記憶體（activation record）配置演進圖，以及 a,c,d 在每一次遞迴的值。（20 分）

```
int foo(int a) {
    int c = 0, d = 0；
    if (a>=100) return a;
    c = a * 10;
    d = foo(c);
    return d;
}
```

【擬答】

編譯器在處理遞迴呼叫時，使用行程記憶體結構中的 Stack Segment 管理每一個副程式的活動紀錄，以下繪製 foo(1)被呼叫後活動紀錄的變化，並將 a, c, d 每次遞迴的值展現在活動紀錄中（假設 foo(1)是在 main()中被呼叫，爲凸顯答案，將 main()的活動紀錄省略）：

foo(1)被呼叫後建立的活動紀錄

欄位	值
Return Value	-
Return Address	main 中呼叫的程式位址
Dynamic Link	main()
Static Link	--
Local Variable: c	0
Local Variable: d	0
Parameter: a	1

下層活動紀錄爲原先產生的活動紀錄（區域變數 c 的值已經改變），當執行到 foo(c)敘述時，呼叫 foo(10)建立上層之活動紀錄

欄位	值
Return Value	-
Return Address	foo(1)中運算式 d = foo(10)的位址
Dynamic Link	foo(1)
Static Link	--
Local Variable: c	0
Local Variable: d	0
Parameter: a	10

欄位	值
Return Value	-
Return Address	main 中呼叫的程式位址
Dynamic Link	main()
Static Link	--
Local Variable: c	10
Local Variable: d	-
Parameter: a	1

foo(100)被呼叫後的活動紀錄

欄位	值
Return Value	-
Return Address	foo(10)中運算式 d = foo(100)的位址
Dynamic Link	foo(10)

Static Link	--
Local Variable: c	0
Local Variable: d	0
Parameter: a	100

欄位	值
Return Value	-
Return Address	foo(1)中運算式 d = foo(10)的位址
Dynamic Link	foo(1)
Static Link	--
Local Variable: c	100
Local Variable: d	-
Parameter: a	10

欄位	值
Return Value	-
Return Address	main 中呼叫的程式位址
Dynamic Link	main()
Static Link	--
Local Variable: c	10
Local Variable: d	-
Parameter: a	1

foo(100)回傳前的活動紀錄

欄位	值
Return Value	100
Return Address	foo(10)中運算式 d = foo(100)的位址
Dynamic Link	foo(10)
Static Link	--
Local Variable: c	0
Local Variable: d	0
Parameter: a	100

欄位	值
Return Value	-
Return Address	foo(1)中運算式 d = foo(10)的位址
Dynamic Link	foo(1)
Static Link	--
Local Variable: c	100
Local Variable: d	-
Parameter: a	10

欄位	值
Return Value	-
Return Address	main 中呼叫的程式位址
Dynamic Link	main()
Static Link	--
Local Variable: c	10
Local Variable: d	-
Parameter: a	1

foo(10)回傳前的活動紀錄

欄位	值
Return Value	100
Return Address	foo(1)中運算式 d = foo(10)的位址
Dynamic Link	foo(1)
Static Link	--
Local Variable: c	100
Local Variable: d	100
Parameter: a	10

欄位	值
Return Value	-
Return Address	main 中呼叫的程式位址
Dynamic Link	main()
Static Link	--
Local Variable: c	10
Local Variable: d	-
Parameter: a	1

foo(1)回傳前的活動紀錄

欄位	值
Return Value	100
Return Address	main 中呼叫的程式位址
Dynamic Link	main()
Static Link	--
Local Variable: c	10
Local Variable: d	100
Parameter: a	1

四、物件導向語言 Java 與 C++最大的不同是 Java class 沒有解構子（destructor）。（每小題 10 分，共 20 分）

(一)請解釋 Java 採用這項設計的背後成因為何？

(二)如果沒有 destructor，請問 Java 如何解決 destructor 原本要解決的問題？

【擬答】

(一)Java 與 C++對於記憶體管理的方式不同，在 Java 中提供垃圾回收（Garbage Collection, GC）機制，透過幾種不同的演算法如「標記-清除（mark-sweep）法」、「停止-複製（stop-copy）法」在記憶體不足時啓動 GC 掃描出沒有被使用的記憶體（稱回記憶體垃圾或懸置物件），將這些垃圾清除甚至重新排序。其主要目的是避免程式設計師不良的記憶體管理習慣，沒有手動釋放記憶體，造成程式執行時所需的記憶體空間被垃圾占滿，另一方面亦能以系統整體最佳化的角度，將有在使用的記憶體重新排序，增加程式執行效能。

(二)在 Java 中，可以使用 null 來移除參考，待移除參考後，系統便會排程執行垃圾回收（Garbage Collection）機制，回收無參考到的物件。

若想加快垃圾回收，可以使用 Java 內建的 System.gc()方法對系統「建議」進行垃圾回收。不過此方法只是提升系統執行回收的優先順序，仍然不一定會直接進行回收。請參考以下程式：

```
Object x = new Object();        //建立物件 x
x = null;                       //將 x 的參考設爲 null，待系統自行釋放
System.gc();                    //提升回收的優先順序
```

另一方面，也可以在 Java 中複寫內建的 finalize()方法，這個方法會在物件被回收的時候自動呼叫。其主要的呼叫目的是如果物件有使用到系統資源（如資料庫連線、檔案資源等），在物件被釋放時若沒有進行資源釋放，則可能使資源繼續被占用，而使其他程式碼無法順利存取資源。

五、下列幾種程式語言是目前 Web programming 中比較當紅的程式語言：Java、javascript、Node.js、PHP、Ruby in Ruby on Rail 及 ASP.net。撰寫 web program 通常要決定使用那一種語言作為 client side 的程式語言以及那一種語言作為 server side 的程式語言。請標示上述的每一種語言，是否可以作為 client side programming，或者是 server side programming，或者是兩者皆是。（20 分）

【擬答】

將 Web Programming 語言的屬性及說明以下表說明之：

語言	執行端	說明
Java	server and client	在 Server 可透過 Servlet 與 jsp 其串接；Client 則有提供 Applet 外掛元件，附加在瀏覽器上。
JavaScript	server and client	在 Server 上使用 Node.js 執行，其特性爲反應迅速，並適合撰寫推播伺服器。
Node.js	server	爲一種在 server 端執行 JavaScript 語言的 framework
PHP	server	
Ruby on Rail	server (and client)	爲一種在 server 端執行 Ruby 語言的 framework，另裡面已內建 prototype（一個 JavaScript 的 framework），支援 Ajax 等前端程式設計。
ASP.net	server (and client)	ASP.NET 是一個已統合的 Web 開發模型，是.NET Framework 的一部分，在撰寫 ASP.NET 應用程式時，可以使用 Microsoft Visual Basic、C#、JScript .NET 和 J#等語言開發。

說明：

Node.js、Ruby on Rail 與 ASP.net 其實都不算是一種語言，而是一種 framework，所以要將之當作語言決定在 Server 端或 Client 端是一件很詭異的事。而其中 Ruby on Rail 與 ASP.net 都是一個網頁應用程式的開發架構，也提供了前端的工具，但在考試中應可暫時忽略。