《程式語言》

試題評析

第一題：考的是講義第四章的範圍，將講義4-4強勢型態定義，加上講義4-3隱性型態轉換(coercion)5-4及課堂講解資料型態相關問題(enum, unionC語言部份)，即可安全過關。

第二題：考的是講義第五章的範圍，5-2中有許多類似例題，只要搭配課堂講解C語言指標和陣列特性(第二章，第五章)，本題就能迎刃而解。

第三題：考的是講義第八章基本題目，難度不高，8-2參數傳遞方式中也有許多相似例題。

第四題：考的是講義第三章基本題目，3-2模糊文法兩大範例的其中一個。

第五題：考的是講義第二章C語言程式設計和6-1C語言運算子的組合題目，需要較強程式設計經驗才能順利回答。

綜合來說，今年題目基本分數落在第二題、第三題、第四題；第一題需要詳細研讀上課講義並且綜合相關章節知識才能完整回答，第五題則爲程式基本功較強的學生才有希望拿到理想分數。一般程度學生可以拿55~65分，有詳細研讀上課內容且有實務撰寫程式經驗的同學則應可拿到75分以上。

一、請回答下列的問題：

（一）解釋什麼是(1)強勢型態程式語言（strongly-typed programming language）和(2)弱勢型態程式語言（weakly-typed programming language）。（10分）

（二）列舉三個理由並舉例說明為何C程式語言不是一個強勢型態程式語言。（10分）

答：

(一)

[強勢型態程式語言]：強勢型態程式語言符合下列要求：

1.每一個變數都靜態地繫結到單一資料型態。
2.變數資料型態繫結後在程式執行過程不能更改其資料型態。
3.某一運算中所用到的兩個變數可以靜態地檢查出它們的相容性。
4.當允許一個變數儲存不同資料型態之值時，其值的型態可以靜態地或動態地檢查出來。
5.不允許編譯器對變數做隱性型態轉換，只能顯示型態轉換。

[弱勢型態程式語言]

1.變數資料型態可以靜態也可以動態決定。
2.變數資料型態在程式執行時期可以更改。
3.可以不檢查運算式中運算元資料型態的相容性。
4.允許編譯器對變數做隱性型態轉換。

(二)

1.C語言對聯合型態變數沒有做實際資料型態的檢查

```
ex. union MyUnion{
        int value;
        char name[20];
    };
    union MyUnion m1;
    strcpy(m1.name， “HelloWorld”);/*m1現在被當作字串型別*/
     int b = m1.value + 3;/*使用者錯誤把m1當作整數型態，編譯器不會檢查此種錯誤 */
```

2.C語言編譯器會做隱性型態轉換(coercion)

ex. double a1 = 1.5;
double a2 = 2.5;
int a3 = 40;
double a4 = a1 + a3;/*應該要寫a4=a1+a2，及兩個浮點數相加，但是寫成a4=a1+a3
編譯器會做隱性型態轉換，將a3轉換成浮點數值去相加*/

3.C語言有可以定列舉資料型態，但是程式可將整數型態實際參數傳給副程式列舉型態變數

ex.

```
enum MyType {Mon， Tue， Wed};/*編譯器分別以整數0， 1， 2代表*/
void f(enum MyType e)/*實際參數值爲整數20，不在此型態定義的範圍，但編譯器不檢查*/
{
    printf("%d"， e);
}
void main()
{
    int a = 20;
    f(a);
}
```

二、假設一個整數佔用四個位元組（4 bytes），考慮一個C程式語言的整數陣列（integer array）intA[4][8][16]，此陣列的起始位址（starting address）為0X22F760，以十六進位（hexadecimal）寫出下列四個printf敘述句（statements）的輸出值（請寫出計算過程）：（每小題5分共20分）

(一)printf("%X\n"， &A[0][1][2]);
(二)printf("%X\n"， &A[0][1][2]+1);
(三)printf("%X\n"， &A[0][1]+2);
(四)printf("%X\n"， &A[0]+3);

答：

int A[4][8][16]

printf("%X\n", &A[0][1][2]);
22F7A8→ 0x22F760 + 4*(0*8*16 + 1*16 + 2)
printf("%X\n", &A[0][1][2] + 1);
22F7AC→ 22F7A8 + 1*4
printf("%X\n", &A[0][1] + 2);
22F820→ 22F760 + 4*(0*8*16 + 1*16 + 2*16)
printf("%X\n", &A[0] + 3);
22FD60→ 22F760 + 4*3*8*16

三、下圖是一個執行時堆疊（run-time stack）中之啟動紀錄（activation record）的示意圖：

Returned value	
Local variables	
Function parameters	
Dynamic link	
Static link	
Return address	

（一）說明如何使用啟動紀錄中的function parameters實作下列兩種副程式的參數傳遞（parameter passing）方法：call-by-value（或稱pass-by-value）和call-by-address（或稱pass-by-address， call-by-reference）。（10分）

（二）考慮下列的C程式語言的程式片段，說明當主程式main呼叫副程式foo之後，副程式foo的啟動紀錄之function parameters內容為何？並寫出主程式main的輸出值。（10分）

```
int c=5;                          int main (void) {
void foo(int x， int* y){              int a=10， b=20;
   int a=1， b=2;                      foo(b， &a);
   *y = a + b* x;                    printf("%d， %d， %d\n"， a， b， c);
   c = a +b +c;                    }
}
```

答：

(一)

[Call by value]：

1.主程式呼叫副程式時，啓動紀錄實例(activation record instance)中形式參數和實際參數有相同大小的記憶體。

2.實際參數的值(記憶體內容)拷貝到對應活動紀錄實例中的形式參數記憶體。

3.程式中任何參考到形式參數的地方，會直接存取到形式參數的記憶體，不會影響實際參數。

[Call by reference]：

1.主程式呼叫副程式時，啓動紀錄實例(activation record instance)中形式參數大小僅爲作業系統存放記憶體位址的大小(以32位元計算機爲例，作業系統以4bytes存放記憶體位址)，而和實際參數大小無關。

2.實際參數的記憶體位址拷貝到對應活動紀錄實例中的形式參數記憶體。

3.程式中任何參考到形式參數的地方，會存取到對應實際參數的記憶體。

(二)1.主程式main呼叫foo之後，foo啓動紀錄中function parameters的內容爲

Function parameters	x = 20
	y = main function中區域變數a的記憶體位址

2.輸出值： 41 20 8

四、考慮下列的BNF法則：

<conditional statement>::= if <condition> then <statement>
| if <condition> then <statement> else <statement>

<statement>::= <assignment statement> |<conditional statement>

(一)假設C1和C2是由<condition>展開的程式碼，S1和S2是由<statement>展開的程式碼，畫出<conditional statement>：

if C1 then if C2 then S1 else S2

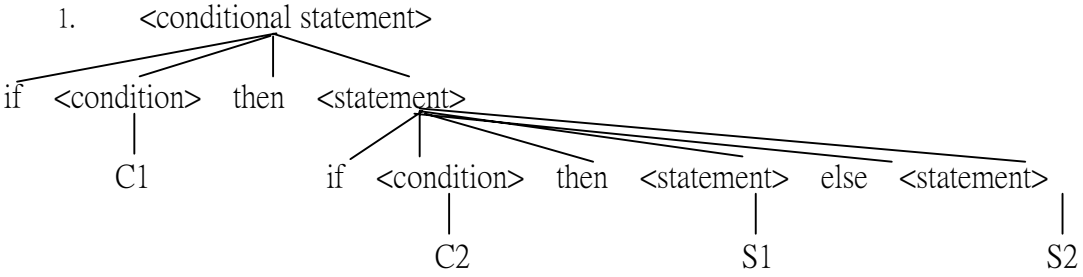
的語法樹（或稱剖析樹，parse tree），並解釋何謂「搖擺else問題」（dangling else problem）。（10分）

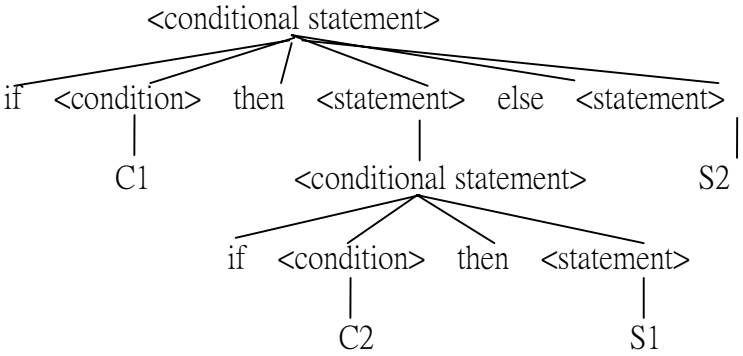
(二)舉出兩個方法，解釋程式語言如何在設計、實作、或使用時解決「搖擺else問題」。（10分）

答：

(一)

1.





2.在如下格式的套疊IF敘述中：

if C1 then if C2 then S1 else S2

無法決定else部分應該屬於第一個if或第二個if，此種情形稱爲懸置else。

(二)解決dangling else問題的兩個方法

1.在語言設計時期修改文法規則，讓else永遠和最近的if成對，例如

<stmt>→<matched_stmt>｜<unmatched_stmt>

<matched_stmt>→if<expr>then<matehed_stmt>else
<matched_stmt>｜<other_stmt>

<unmatched_stmt>→if<expr>then<stmt>
｜if<expr>then<matched_stmt>else
<unmatched_stmt>

2.利用特殊區分符號來決定else所對應的if，以C語言爲例，利用成對大括號{}來決定else所對應的if，例如以if C1 then if C2 then S1 else S2，若else是對應C1的if，則以C語言撰寫如下：

```
if (C1) {
    if(C2) {
        S1
    }
} else {
    S2
}
```

若else是對應C2的if，則以C語言撰寫如下

```
if(C1) {
    if(C2) {
        S1
    } else {
        S2
    }
}
```

五、考慮C程式語言的位元運算（bitwise operation），變數m和陣列（array）n的宣告如下：

```
unsigned int m;
unsigned char n[4];
```

假設m的二進位值（binary value）為：

$b_{32}b_{31}b_{30}b_{29}b_{28}b_{27}b_{26}b_{25}b_{24}b_{23}b_{22}b_{21}b_{20}b_{19}b_{18}b_{17}b_{16}b_{15}b_{14}b_{13}b_{12}b_{11}b_{10}b_9b_8b_7b_6b_5b_4b_3b_2b_1$

寫一個C語言的程式將陣列n的元素（element）設定為：

n[0]： $b_{31}b_{32}b_{29}b_{30}b_{27}b_{28}b_{25}b_{26}$

n[1]： $b_{23}b_{24}b_{21}b_{22}b_{19}b_{20}b_{17}b_{18}$

n[2]： $b_{15}b_{16}b_{13}b_{14}b_{11}b_{12}b_9b_{10}$

n[3]： $b_7b_8b_5b_6b_3b_4b_1b_2$

即是將m的二進位值，以每兩個位元一組，作位元調換（bit swap），再切割成四個位元組。除了迴圈控制變數（loop control variable）外，程式中不可使用+， －， *， /， %的算術運算（arithmetic operations）（可以宣告和使用其他變數）。（20分）

答：

```
#include <stdio.h>

void main()
{
    const unsigned int UN_RIGHT_SHIFT_MASK = 0x55555555;/*binary 01010101…01*/
    const unsigned int UN_LEFT_SHIFT_MASK = 0xaaaaaaaa;/*binary   10101010…10*/
    const unsigned int UN_BYTE_MASK = 0x000000ff;/*binary 00…000011111111*/
    int BitShiftSet[] = {24,   16,   8,   0};
    unsigned int m = 0x12345678;
    unsigned char n[4];
```

```
    /*Move current value 1 bit left and 1 bit right to do bit shift*/
    unsigned int unRightShiftNum = (m >> 1) & UN_RIGHT_SHIFT_MASK;
    unsigned int unLeftShiftNum = (m << 1) & UN_LEFT_SHIFT_MASK;
    unsigned int unSwithedNum = unRightShiftNum | unLeftShiftNum;

    printf("%X\n",    unSwithedNum);
    /*Assignd the proper bit range to array n by moving the bit range to
      the start and clear value with bit index >= 8*/
    for (int i = 0; i < 4; ++i) {
        n[i] = unSwithedNum >> BitShiftSet[i] & UN_BYTE_MASK;
        printf("%X\n",    (unsigned int)n[i]);
    }
}
```