《程式語言》

一、請回答下列程式語言的問題：

（一）就程式語言來說，什麼是Object-Oriented languages? Functional languages? Logic languages? Script languages?（12分）

（二）從上述（一）的四種語言中，除Logic languages外，每種語言各舉兩個現存的語言。（6分）

（三）最近語言的發展中，像Functional languages與Script languages，都逐漸引進物件導向性質（OO），就這兩類語言，各舉一個近來發展出且具有OO性質的語言。（2分）

試題評析	此題為程式語言分類的典型考題，屬於程式語言概論的題目，與101年檢事官題目極為相似。對於掌握考古題的考生，應該很容易可以拿到基本分，唯第三小題需要能回答出具有物件導向特性的函數型、腳本語言，需要實務經驗才能作答。
考點命中	1.《高點102年程式語言講義第六回》，金乃傑老師編撰，頁55-56。 2.《高點102年程式語言講義總複習第一回》，金乃傑老師編撰，考點30。

答：

(一)1.Object-Oriented language：物件導向語言，由類別（class）實作的物件（objects）所構成，物件間透過訊息傳遞溝通操作。物件導向語言必具有三大特性，說明如下：

(1)封裝（Encapsulation）：將資料及操作包在類別中，有助於資訊隱藏（Information hiding）、增加可讀性、安全性。

(2)繼承（Inheritance）：父類別的資料和操作可供子類別使用，增加再用性（reusability）、減少重複程式碼撰寫，加速開發、修改、降低錯誤率、維護成本。

(3)動態繫結（Dynamic Binding）：執行時期動態的決定一個副程式呼叫所要執行的副程式，達成多型（Polymorphism），可以提高執行彈性高、減少程式判斷與重複撰寫。

2.Functional language：函數型語言，其程式的基本單位為運算式，運算式由函數呼叫指令所構成。其語法特色如下：

(1)操作變數時不使用指定敘述，資料由函數傳入之參數處理。

(2)由函數遞迴呼叫完成重複結構。

(3)具有參考透明性（Referential Transparency），函數給予相同的參數，就會有相同的結果，不會有邊際效應。

3.Logic language：邏輯型語言，如Prolog，程式由一群宣告所組成，而不是指定敘述或控制流程。語言透過以符號邏輯（symbolic logic）的形式來描述程式，其主要目的為推論。程式可以分為事實、規則與詢問子句三大類。事實提供程式推論的基礎，規則則是協助程式判斷true或false的依據。邏輯行程式語言主要提供true or false的推論，或是變數求值推導，專長為邏輯符號的處理，算術運算效能低，為人工智慧的高階語言（第五代電腦語言）。

4.Script language：腳本語言，此種語言通常有兩種執行模式，使用者可以直接在直譯器中輸入指令碼使之執行，也可以將指令碼寫在純文字文件中，讓直譯器讀取並直接執行。腳本語言原為系統管理員建立便捷自動的管理工具，隨著時代逐漸發展，功能愈趨強大，也提供物件導向等功能。

(二)語言實例如下：

1.Object-Oriented language：如C++、Java、C#、Smalltalk、Ada等。

2.Functional language：如Lisp、Scheme、APL、SNOBOL等。

3.Script language：如PHP、JavaScript、Bash、ActionScript、Python、Perl等。

(三)有物件導向性質的Functional與Script language如下：

1.Functional language：如OCaml、Extensible ML、O'Haskell、Scala、R語言

2.Script language：如PHP、JavaScript、Perl、Python、Ruby

二、考慮下述用類似C語言的語法寫出的程式：

```
Void swap(int a, int b) {
   int temp;
   temp = a;
   a = b;
   b = temp;}
void main() {
   int value = 1, list[4] = {2, 3, 4, 5}  //Array list is list[1..4]
   swap(value, list[1]);
   swap(list[1], list[2]);
   swap(value, list[value])}
```

根據以下的參數傳遞（parameter passings）模式，逐一列出在程式執行完畢之後value與list值：

(一)Passed by value?（5分）

(二)Passed by reference?（5分）

(三)Passed by name?（5分）

試題評析	此題爲典型的參數傳遞方式考題，屬於99年警察、94年地特的相似題。但此題出現多年未見的passed-by-name，對許多考生而言可能較爲陌生。不過就班內學生而言，若有充分做練習題，應還是能游刃有餘的作答。
考點命中	1.《高點102年程式語言講義第二回》，金乃傑老師編撰，頁4-7。 2.《高點102年程式語言講義總複習第一回》，金乃傑老師編撰，考點10。

答：

(一)Passed by value執行完後，value與list之值如下：

value	list[1]	list[2]	list[3]	list[4]
1	2	3	4	5

(二)Passed by reference

value	list[1]	list[2]	list[3]	list[4]
1	3	2	4	5

(三)Passed by name

value	list[1]	list[2]	list[3]	list[4]
1	2	1	4	5

解析：

使用passed by value時，副程式swap所交換的數值與主程式使用不同的記憶體空間，因此副程式的變化不會影響主程式變數的數值。

使用passed by reference時，副程式swap所交換的數值與主程式使相同的記憶體空間，因此會執行主程式中變數交換。

使用passed by name時，對副程式中形式參數做名詞取代，前兩次交換與passed by reference結果相同，但第三次有牽涉陣列，故副程式中敘述變爲：

```
temp = value;
value = list[value];
list[value] = temp;
```


由於執行過第二次交換後value = 2；list[value] = 1，故帶入結果如下：

```
temp = value = 2;
value = list[value = 2] = 1;
list[value = 1] = temp = 2;
```

三、根據每一個元素（element）包含16個英文字母的circular queue，當用長度m的陣列（array）來實作使用JAVA語言寫出下列兩個方法的完整定義：

(一)Enqueue（10分）

(二)Dequeue（10分）

試題評析	此題爲實作資料結構中的circular queue，由於題目指定使用Java，故必須用物件導向方式撰寫。值得注意的是circular queue用陣列實作時，會爲了判斷queue是否爲空，因此尾端會空下一格，撰寫時必須要小心檢查才能使queue如預期的運作。
考點命中	1.《高點102年程式語言講義第四回》，金乃傑老師編撰，頁6-15。 2.《高點102年程式語言講義總複習第一回》，金乃傑老師編撰，考點16。

答：

以Java撰寫如下：

```
3  public class CircularQueue {
4      char[][] queue;                          //儲存circular queue的陣列空間
5      int limit, front, rear;
6
7      CircularQueue(int m){
8          queue = new char[m][16];             //初始化每個元素16個字元
9          limit = m;                           //初始化上限
10         front = rear = 0;                    //初始化頭尾指標
11     }
12     //Enqueue方法
13     void Enqueue(char[] data) throws Exception{
14         int new_rear = (rear+1)%(limit);//產生新的尾指標(考慮循環)
15         if(new_rear == front) throw new Exception("full");
16                                              //陣列滿了彈出例外(善用Java例外功能)
17         queue[rear] = data;                  //將資料插入陣列
18         rear = new_rear;                     //移動尾指標
19     }
20     //Dequeue方法
21     char[] Dequeue() throws Exception{
22         int new_front = (front+1)%(limit);//產生新的頭指標(考慮循環)
23         if(front == rear) throw new Exception("empty");
24                                              //陣列空了彈出例外(善用Java例外功能)
25         char[] data = queue[front];          //取出陣列資料
26         front = new_front;                   //移動首指標
27         return data;                         //傳回queue內容
28     }
29
30     public static void main(String args[]){ //測試程式(考試時可不寫)
31         CircularQueue cq = new CircularQueue(10);
32         for(int i = 0; i < 10; i++){
33             char data[] = {Character.forDigit(i, 10)};
34             try{
35                 cq.Enqueue(data);
36             }catch(Exception e){System.out.println(e.toString());}
37         }
38         for(int i = 0; i < 10; i++){
39             try{
40                 System.out.println(cq.Dequeue());
41             }catch(Exception e){System.out.println(e.toString());}
42         }
43     }
44 }
```

四、對C#與JAVA的concurrency而言：

(一)C#的thread可以是actor thread嗎？JAVA的呢？（5分）

(二)C#的thread可以被非同步呼叫嗎？JAVA的呢？（5分）

(三)C#的sleep method與JAVA的sleep method有何不同？（5分）

試題評析	此題爲並行程式考題，以Java的並行爲主，並比較C#。由於國考以往不常以C#出題，因此必須搭配C#相關實務經驗否則不易回答。
考點命中	1.《高點102年程式語言講義第三回》，金乃傑老師編撰，頁84-90。 2.《高點102年程式語言講義總複習第一回》，金乃傑老師編撰，考點14、15。

答：

(一)actor是並行運算模型中參與者模式（Actor Model）上的角色。actor是一種程式上的抽象概念，並行運算的基本單元，每一個actor擁有自己的狀態及行爲，這些狀態及行爲都被封裝在actor中。其他的actor只能夠透過訊息傳遞的方式來與這個actor溝通，當actor接收外部訊息時，這些訊息會被放在message queue裡依序處理，actor可以做出一些反映，如建立更多的actor、傳送更多的訊息。Actor Model中訊息傳遞與一般的訊息傳遞或函數呼叫最大的不同是送訊息的及處理訊息的機制是非同步的，因此訊息不一定會立刻被處理，甚至傳遞後不一定會收到回應。在C#與Java中，thread是程式最小的執行單位，而多執行緒並行程式取得鎖（lock，在C#中叫Monitor，Java中叫mutex）時，必須要等到鎖被釋放掉才能取得，此即爲actor的形式（呼叫不會馬上處理）。故在C#與Java中，都可以有actor thread。

(二)非同步呼叫（Asynchronous method invocation）指的是在進行函數呼叫（或方法method訊息傳遞）時，不等函數傳回值便繼續執行後面的敘述。非同步呼叫通常用於預期會執行很久的函數，爲了加快程式的反應，因此讓程式等到函數執行完的「事件」啓動時，再接收函數的處理結果。C#與Java都提供非同步呼叫：

1.C#透過IAsyncResult、Event Based的非同步模型實作。

2.Java透過java.util.concurrent.FutureTask類別，建立event來處理。

(三) Thread的Sleep方法可以讓Thread暫停指定毫秒或直到被中斷。C#與Java的sleep語法如下：

1.C#：Sleep(int millisecondTimeout)或Sleep(System.TimeSpan)

2.Java：sleep(long millis)或sleep(long millis, int nanos)

除了方法大小寫外，主要差異如下：

1.在Java中，使用sleep()會使執行緒進入TIMED_WAITING的狀態，但此時若該執行緒持有鎖（mutex），則鎖不會放掉，意味著如果此執行緒有其他synchronized的方法，也會被鎖住不能執行。但C#使用Sleep()時會進入WaitSleepJoin狀態，此時是不持有鎖（在C#中叫monitor）的，因此C#的Sleep(0)可以充當Java的yield()用，代表退出執行狀態，讓其他執行緒進入排班。

2.Java的sleep第二組參數爲奈秒，若系統支援下，可以更精確的控制執行是睡眠的時間；C#則提供系統時間的參數，可以直接使用TimeSpan插入時間間格，方便計算。

3.當Java進入TIMED_WAITING狀態時，可能會被其他執行緒插斷（Interrupted），插斷時就會拋出InterruptedException，因此Java的sleep method一定要包在try-catch中；但C#則沒有要強制將Sleep包在try-catch中。

五、考慮下述Prolog程式：

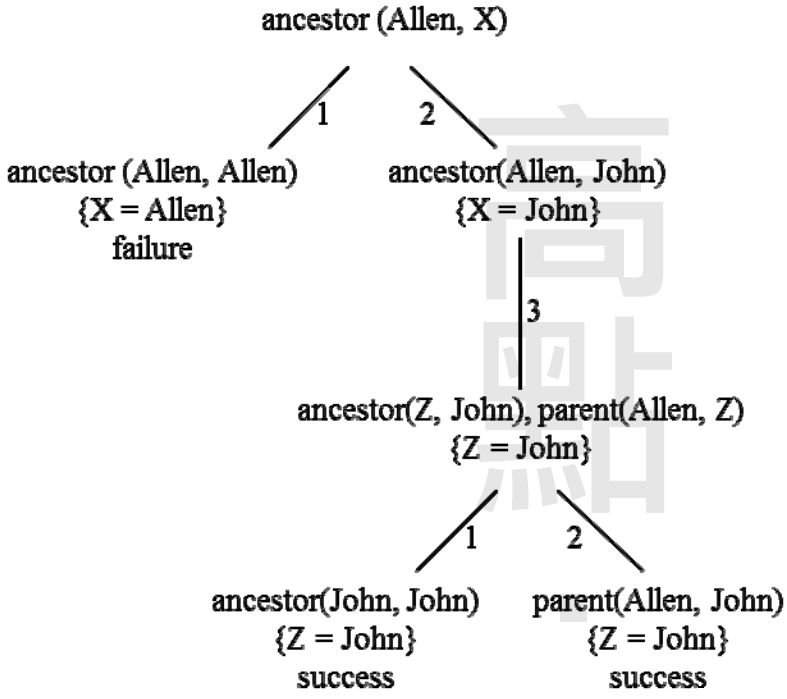
```
ancestor (x, x).
ancestor(x, y) :- ancestor(z, y), parent(x, z)
parent(Allen, John).
```

用一個subgoal的搜尋樹（search tree），描述此Prolog程式在執行查詢：ancestor(Allen, X)時的執行步驟。（10分）

試題評析	此題爲邏輯型程式語言考題，屬於100關務的相似題，不過在近年考試中屬於罕見題型。此題要求考生畫出推導邏輯，必須要掌握規則由上而下由左而右依序檢查的順序。由於邏輯並不複雜，因此只要充分準備的考生，應該還是能寫出正確答案。
考點命中	1.《高點102年程式語言講義第五回》，金乃傑老師編撰，頁9-11。 2.《高點102年程式語言講義總複習第一回》，金乃傑老師編撰，考點21。

答：

subgoal搜尋樹如下圖所示：



最後在Z值爲John時找到X值爲John，得解。

六、考慮下述的Scheme程式：

```
(define A
(lambda ()
   (let* ((x 2)
          (C (lambda (P)
               (let ((x 4))
                 (P))))
          ( D (lambda ()
                x))
          (B (lambda ()
               (let ((x3))
                 (C D)))))
         (B))))
```

(一)這程式列印什麼東西？(10分)

(二)如果Scheme使用dynamic scoping和shallow binding，它會印出什麼？(10分)

試題評析	此題牽涉領域、函數型程式語言，屬於複雜度較高的題目，若有觀念沒有釐清，相當有可能出錯。 第一小題求scheme語言的輸出值，不過題目存在瑕疵，因爲在題目中僅做函數宣告而已，並未執行函數A，所以不會有資料印出，儘可能在某些直譯器秀出A函數建立完成的訊息。但推測題意，應自行加入呼叫語法，不然無法完成考試需求。 第二小題求動態領域法兩種實作方式的結果，也是初次出現在三等考試中的題目，考生必須要有相當深厚的程式基礎來能作答。

考點命中	1.《高點102年程式語言講義第五回》，金乃傑老師編撰，頁3-6。 2.《高點102年程式語言講義第三回》，金乃傑老師編撰，頁41-46及上課補充。 3.《高點102年程式語言講義總複習第一回》，金乃傑老師編撰，考點1, 22。

答：

(一) 若使用者呼叫A，則印出：

2

(二)

若使用者呼叫A，

使用dynamic scoping，匿名函數中所用到的變數值是以該函數第一次被呼叫時的領域爲準，故會印出：

3

（因爲D在B中被呼叫，所以參考到B中的x）

使用shallow binding，匿名函數中所用到的變數會先參考被呼叫的環境，故會印出：

4

（因爲D在C中被執行，所以參考到C中的x）

解析：

let函數：定義函數裡的區域變數，例如(let ((x 5)) (* x 3))會將x定義爲5，並帶入後面的(* x 3)所以計算出15。

題目用仿C語言可以表示爲：

```
function A{
        x = 2;
        C = lambda(P){
                x = 4;
                return P;
        }
        D = lambda(){
                return x;
        }
        B = lambda(){
                x = 3
                return C(D);
        }
        return B();
}
```

另外，實際上dynamic scoping包括deep binding與shallow binding兩種，故題意有模糊的地方，推測應爲比較deep binding與shallow binding之結果，故擬答中dynamic scoping是套用deep binding的答案。