《程式語言》

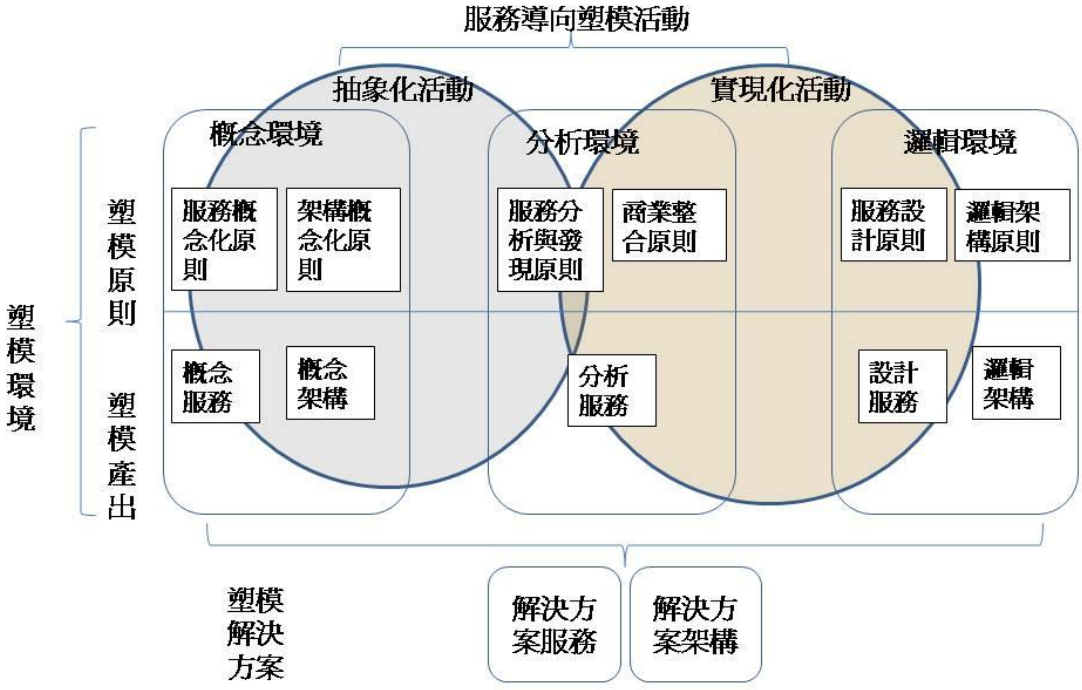
試題評析

今年程式語言考題分析如下：
第一題：相當新的開發模式，須要有實務經驗或有廣泛涉獵相關知識才能解答完整。
第二題：本題範圍屬於基本題目，請見班內講義第二、三、四章、函數型語言補充講義和上課補充內容。
第三題：班內講義第十章題目，是並行副程式中的一種，依照題目提供fork爲process並行的概念(非執行緒)以及上課講解執行緒和行程的差異，難度不高。
第四題：班內講義第三章基本題目，講義及課堂講解中有相同題目。
第五題：班內講義第六章基本題目，講義及課堂講解中有相同題目。
綜合來說，今年題目基本分數落在第二題、第四題、第五題，對學生來說應該不難回答，唯第三題需要充分了解課堂講解process和thread的差異和副程式並行概念，屬於中等難度題型，第一題則是涉獵廣泛領域的學生才能拿到分數。一般程度學生可以拿60~70分，詳細研讀上課內容且有廣泛知識的學生，則可拿到70分以上。

一、服務導向架構（Service-oriented architecture, SOA）是一種事件驅動（Event-driven）的程式結構（Programming paradigms），試畫出服務導向建模框架（Service-Oriented Modeling Framework），並以此框架說明SOA之優點。（20分）

答：

（一）服務導向建模框架(Software oriented modeling framework, SOMA)



（二）服務導向建模框架(Software oriented modeling framework, SOMA)是一種軟體開發模式，定義以服務爲導向的軟體開發架構需要的原則和相關的塑模工具，目的是爲了讓企業或組織建構一個具有彈性、可重複使用的整合性架構，一個企業業務的應用程式稱爲一個獨立的「邏輯單位」，而對企業營運層面而言，則可稱爲是一項「服務」，在企業的整體運算環境中，就存在著多個「獨立邏輯／業務服務」，且需要對

其進行妥善設計、開發、佈建、管理等，也因此需要採行服務導向架構。
服務的核心概念底層在將軟體轉化爲可重複使用與分享的服務，因此，架構的建立，首先在分析各個應用系統中，有哪些功能是必要開放成爲服務的；一般直覺的方法就是看看有哪些功能重複出現在不同的系統中，而據此決定哪些功能是要包成服務。而另一種方式，則是從流程中著手，藉由 IT 與 Business 部門的合作，進行流程內涵的審視與分析，一方面由流程實務需求上，確定哪些功能需要被建構成服務，另一方面可藉由審視流程的機會，進行流程的改進。
優點：
1.更具生產力與彈性的應用程式：
應用軟體是以各種服務組合而成，可隨時以新版本的服務替換掉舊的服務，讓軟體更新更有彈性。
2.更快速且更具成本效益的應用程式發展：
新開發的應用軟體可以用現成已開發的各種服務組合而成，不須重新開發。
3.更具管理性及安全性的應用程式：
SOMF定義了詳細以服務爲導向的開發原則，其中應用程式透過服務存取資料，可以在存取資料的服務上加強安全設計與控管。

二、請指出下列敘述為“真”或為“假”，並說明之。（20分）
(一)C++語言的識別符號（identifiers）是context-free。
(二)有限語言（finite language）都是regular。
(三)函數型語言（Functional languages）是基於lambda calculus。
(四)++(x+y)的錯誤可由編譯程式（compiler）中的語法分析程式（syntax analyzer）偵測得知。

答：
(一)假，以變數名稱爲例，在同一個區塊中，上文宣告過的變數識別字不能再宣告一次，故宣告變數所使用的識別字是與上下文有關(context-sensitive)。

Ex.
```
{
   int i;//識別字i已經使用
  …
  int i;//不能再使用識別字i來宣告變數
}
```

(二)真，Finite language是regular language的子集。Regular language沒有限制語言中字彙的數目而Finite language則是有限字彙。
(三)真，lambda calculus定義無名稱函數計算的語法及語意，是函數型語言中函數定義及運算的基礎。
(四)真，++(x + y)，其中++運算子所需的運算元必須是變數的左值(位址)，而x+y算出的是一個數值，故在編譯時期的語法分析程式就能找出。

三、下列是UNIX系統以process為單位的並行處理方式，其中fork()是用來產生另一個process，請依序寫出其執行結果。（20分）
```
#include <stdio.h>
int sum;
void main() {
      sum = 0;
      fork();
      for(int j = 0; j < 4; j++) {
          printf("j = %d\n", j);
          fflush(stdout);
```

```
            sum += j;
        }
        printf("sum = %d\n",sum);
        exit(0);
    }
```

答：

利用fork來產生新的process會有和原來process獨立的資料(Data)空間，包含全域變數和區域變數，並且子process(被fork產生的process)和父process(呼叫fork的process)。依照作業系統的排班演算法來安排執行順序可能的執行結果(依作業系統排班演算法而定)：

j = 0
j = 1
j = 0
j = 1
j = 2
j = 3
j = 2
j = 3
sum = 6
sum = 6

四、請重寫下述文法使其成為right linear（右側遞迴）文法。（20分）

<E> → <T> | <E> * <T>
<T> → <V> | <V> + <T>
<V> → a

答：

<E>→<T><E`>
<E`>→*<T><E`>
| ∈
<T>→<V><T`>
<T`>→+<V><T`>
| ∈
<V>→a

五、(一)何謂捷徑計算（short-circuit evaluation）？（10分）
(二)若語言本身不提供捷徑計算，則下列程式片段會出現什麼錯誤訊息？（10分）

```
index := 1;
while(index <= listlen) and (list[index] <> key) do index := index + 1;
```

（假設list[1..listlen]為被查詢之陣列，而key為要查詢之值）

答：

(1)運算式的結果無需計算所有的運算元或運算子即可決定的計算方式。
(2)陣列存取超過陣列宣告範圍的錯誤，當index值等於listlen + 1時，index <= listlen判斷式結果爲false，但是由於不提供捷徑運算，所以list[index] <> key的判斷式仍然會執行，此時index = listlen + 1，超過陣列可存取範圍。