《程式語言》

試題總評析	本份考卷之難易分際在於如何透過程式語言的理論概念去了解即整合實務上的情況(像是第1題、第5題)，實務上的情況則偏重在C語言的整合比較，對於熟悉C語言功能者應能獲得較佳的成績。五個題目的範圍，除了第4題須有資料結構之能力外，其餘均為程式語言標準範疇概念，一般程度考生在五個題目中，應能掌握部份基本題目的分數，大約是50~60分。

一、(一)試說明一個好的程式語言（programming language）應具備之重要特徵。（10分）
(二)請以上述所述之特徵評斷C語言之優缺點。（10分）

答題關鍵	了解C語言的功能，其與C++之間的差異。
試題評析	本題在第二小題部分有極明確的分類，重點在於第二小題必須整合了解C語言在各項特徵上的支援能力，也因此使題目創造了部分整合的難度。
參考資料	Concepts of Programming Languages, 5th edition, Robert W. Sebesta
高分閱讀	高點胡世雄《程式語言》第一章，p 1-11

【擬答】

(一)

一個好的程式語言應具備四的基本特徵：

1. **可讀性**(readability)**：容易閱讀與容易了解的特性。包括有**

   甲、 簡單性及正交性

   只具備少數的的基本結構及簡單的組合規則，並且每一種可能的組合都是合法且有意義的，此即滿足正交性(orthogonality)。

   語言在設計上愈具有正交性，則語言的規則中愈少有例外情況，因此易學、易讀、易了解，此即滿足簡單性(simplicity)。

   乙、 控制結構

   表示使用的語言是否支援結構化程式設計。

   丙、 資料型態與結構

   表示使用的語言所支援的資料型態為何？以及可以建立哪些資料結構。

   丁、 語法考量

   指程式語言使用之識別字格式，及提供哪些關鍵字，能否從語法格式了解其意義之能力。

2. **可寫性**(writability)**：容易建立程式的特性。**

   甲、 簡單性與正交性

   乙、 支援抽象化

   指程式語言允許使用者定義複雜的結構和運算，並且使用他們時可忽略掉無關細節之能力。可分為程序抽象化和資料抽象化。

   丙、 可表達性

   指程式語言可以表達各種運算的能力和特性。

3. **可靠性**(reliability)**：語言的設計可讓使用者不易犯下錯誤，即使犯錯也可以容易找出錯誤的特性。**

   甲、 型態檢查

   表示程式語言在編譯和執行時期具備進行型態檢查的能力。

   乙、 例外處理

   表示程式語言可以偵測例外及處理例外狀況的能力。

   丙、 別名的使用

   表示程式語言限制使用別名的能力。

4. **成本**(cost)**：設計時考量開發相關成本降低的特性，如下**

   甲、 學習成本

   乙、 撰寫程式之成本

   丙、 編譯成本

丁、 執行成本

戊、 編譯程式之建構成本

己、 低可靠性之成本

庚、 維護程式之成本

(二)

影響因素	C語言
簡單性及正交性	●
控制結構	●
資料型態與結構	●
語法考量	●
支援抽象化	●
可表達性	●
型態檢查	○，允許函數中的參數不作型態檢查
例外處理	C++提供
別名的使用	●

●提供，○部分功能

二、(一)請證明下述文法是混淆的（ambiguous）。（10分）

<S>→<A>

<A>→<id>｜<A>+<A>

<id>→a｜b｜c

(二)下述句子（1.到5.）有那些可由上述文法產生？（10分）

1.a+b+c

2.a+a+a

3.a

4.b

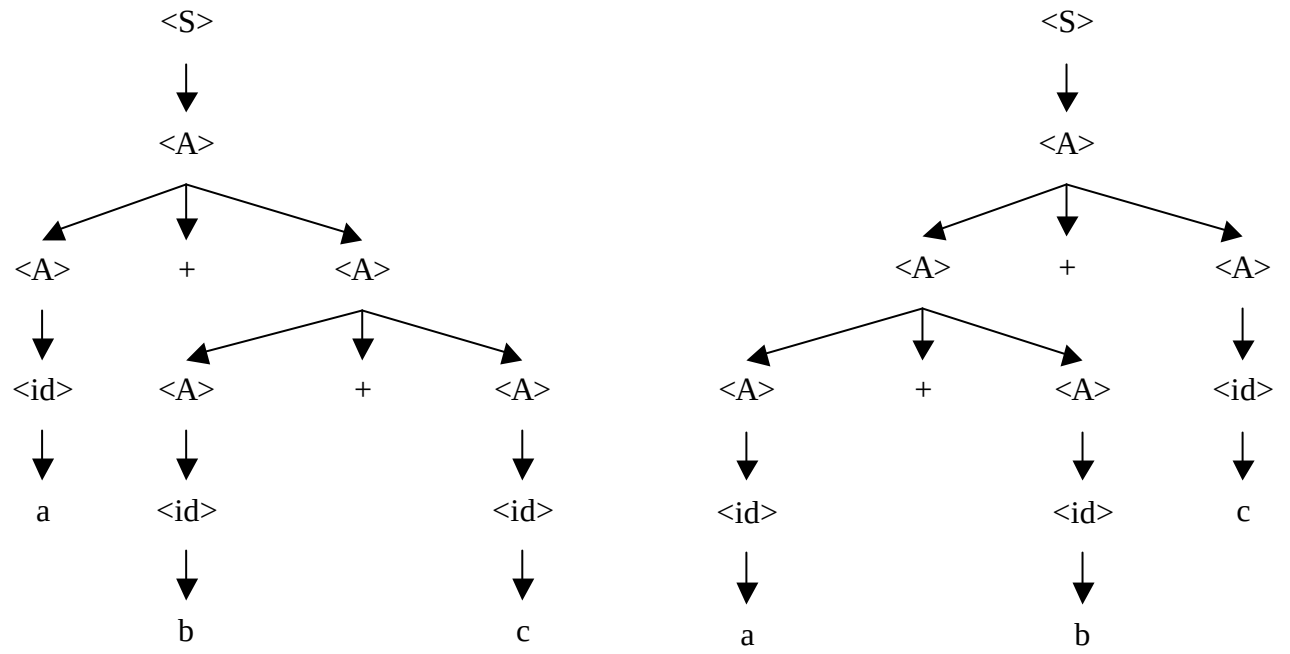
5.c

答題關鍵	了解混淆語法的定義，利用剖析數來解釋問題。
試題評析	考題以BNF為主討論語法，難易適中。
參考資料	Concepts of Programming Languages, 5th edition, Robert W. Sebesta
高分閱讀	高點胡世雄《程式語言》第二章，p 2-11

【擬答】

(一)

如果一語言的某一語句，依其bnf文法規則來推倒可會出二個或二個以上不同的剖析樹，則稱該語言的文法為不明確文法，考慮 a+b+c



(二)
以上皆可。

三、考慮下列程式片段：

```
main(){
 int i, a[2]；
 i=1; a[1] =12; a[2] =6；
 p(i, a[i]);
 write(i, a[1], a[2]);
 p(a[i],i);
 write(i, a[1], a[2]);
}
p(int x, int y){
 x=x+1
 y=y+1
 write(x,y);
 x=x−1;
 y=y−1;
}
```

(一)若參數傳遞為以值傳遞（passed by value**），則所有印出（**write**）之結果為何？（10分）**
(二)若參數傳遞為以址傳遞（passed by address**），則所有印出（**write**）之結果為何？（10分）**

答題關鍵	**本題答題關鍵在於傳址呼叫時副程式的形式參數會與主程式之實際參數共用相同的記憶體，而當**p(i,a[i])**執行時，**x**指向**i**的記憶體，**y**指向**a[i]**的記憶體，而當**x=x+1**執行後，**y**所指記憶體便成為**a[2]**，因此會進行**write(2, 6)
試題評析	**本考題在第一小題，屬於基本概念題目，第二小題則需要詳細了解記憶體參照的概念方能順利獲得分數**
高分閱讀	**高點胡世雄《程式語言》第七章，**p.7-5~7-6

【擬答】

(一)傳值呼叫為in mode**，主程式除實際參數之外會為形式參數建立所需記憶體位置。**

	i	a[1]	a[2]	x	y
i=1; a[1] =12; a[2] =6；	1	12	6		
p(i, a[i]);	1	12	6	1	12
x=x+1;y=y+1;	1	12	6	**1**	**12**

write(x,y)**即write(2,13)**	1	12	6	2	13
x=x-1;y=y-1;	1	12	6	1	12
write(i, a[1], a[2])**即write(1,12,6)**	**1**	**12**	**6**	1	12
p(a[i],i);	1	12	6	12	1
x=x+1;y=y+1;	1	12	6	13	2
write(x,y)**即write(13,2)**	1	12	6	**13**	**2**
x=x-1;y=y-1;	1	12	6	12	1
write(i, a[1], a[2])**即write(1,12,6)**	1	12	6	12	1

(二) 傳址呼叫為inout mode**，主程式並不會為副程式之形式參數建立所需記憶體位置，而是與主程式之實際參數共用記憶體。**

	i	x	a[1]	y	a[2]	
i=1; a[1]=12; a[2]=6**；**	1		12		6	
p(i, a[i]);	1		12		6	
x=x+1;y=y+1;	2		13		6	
			a[1]		a[2]	y
write(x,y)**即write(2,6)**	**2**		**13**		**6**	
x=x-1;y=y-1;	1		12		6	
write(i, a[1], a[2])**即write(1,12,6)**	**1**		**12**		**6**	
	i	y	a[1]	x	a[2]	
p(a[i],i);	1		12		6	
x=x+1;y=y+1;	2		13		6	
write(x,y)**即write(13,2)**	**2**		**13**		6	
x=x-1;y=y-1;	1		12		6	
write(i, a[1], a[2])**即write(1,12,6)**	1		12		6	

四、請以任一種程式語言（或虛擬語法）寫出一資料抽象型態（data abstract**）之堆疊（**stack**）結構，另至少必須包含有**initialization, push, **及**pop**等運算。（20分）**

答題關鍵	**本題需了解堆疊之概念，再輔以資料抽象型態的定義方能順利答題。**
試題評析	**試題單就程式語言來說，資料抽象型態的概念是適中的方向，不過實作堆疊則需要資料結構功力的輔助，才能獲致高分。**
高分閱讀	**高點胡世雄《程式語言》第九章，**p 9-**11**

【擬答】

利用Module-2**的資料抽象型態建構**STACK**之**module。

(一)首先定義部分：

```
DEFINITION MODULE stackmod;
  TYPE stacktype;
  PROCEDURE empty(s:stacktype)：BOOLEAN;
  PROCEDURE push(VAR s:stacktype; element:INTEGER);
  PROCEDURE pop(VAR s:stacktype):INTEGER;
  PROCEDURE initialize(VAR s:stacktype);
END stackmod.
```

(二)實作部分：

```
IMPLEMENTATION MODULE stackmod;
  FROM InOut IMPORT WriteString, WriteLn;
  FROM Storage IMPORT NEW;
  CONST max=100;
  TYPE stacktype = POINTER TO RECORD
        List:ARRAY[1..max] OF INTEGER;
        Topsub：[0..max];
        END;
PROCEDURE empty(s:stacktype)：BOOLEAN;
  BEGIN
  RETURN s^.topsub=0;
  END empty;
PROCEDURE push(VAR s:stacktype; element:INTEGER);
```

```
  BEGIN
  IF s^.topsub=max THEN
        WriteString("ERROR-Stack overflow");
        WriteLn;
  ELSE
        s^.topsub:=s^.topsub+1;
        s^.list[s^.topsub]:=element;
  END
END push;
PROCEDURE pop(VAR s:stacktype);
  BEGIN
  IF empty(s) THEN
        WriteString("ERROR-Stack underflow");
        WriteLn;
  ELSE
        s^.topsub:=s^.topsub-1;
  END(*of IF empty ..*)
END pop;
PROCEDURE top(s:stacktype):INTEGER;
  BEGIN
  IF empty(s) THEN
        WriteString("ERROR-Stack underflow");
        WriteLn;
  ELSE
        RETURN s^.list[s^.topsub];
  END(*of IF empty ..*)
END top;
PROCEDURE initialize(VAR s:stacktype);
  BEGIN
  NEW(s);
  s^.topsub:=0
  END initialize;
END stackmod.
```

五、(一)C**程式語言可以用來設計多工作業系統**（multi-process OS）**，然而它並不被認為是一個平行語言**（concurrent programming language）**，試說明其理由。**（10**分）**

(二)**試說明使用**monitor**或**message passing**類型之平行語言**（concurrent programming language）**之優點（和使用**semaphore**作比較）。**（10**分）**

答題關鍵	**了解平行語言具備之特徵，透過其特徵可以檢証C為何不構成平行語言的要件，另外需了解semaphore的缺失，從而能比較出monitor與message如何解決semaphore之缺失。**
試題評析	**考題除注重理論的比較，上能輔以實務之分析，有提升考生水平之意義，要或高分須有良好的整合能力。**
高分閱讀	**高點胡世雄《程式語言》第十章 p 10-13**

【擬答】

(一)

程式語言的並行性是指程式中可提供多線執行的控制流程，所謂執行線(thread)是指程式的控制流可到達的點之序列，因此程式語言支援並行性極事允許有多條執行線同時在程式中執行。

支援並行性的程式語言，必須提供六個功能

一、 有方法描述哪一段程式碼可以與其他程式碼平行的執行，此段程式碼可以是：

甲、 處理程序

乙、 工作

丙、 並行子句

二、 能夠啟動並行程式單元之執行

三、 能夠提供一種機制，以保證共享資料之互斥使用

四、 能夠提供並行單元單元之間的同步

五、 必須有方法可以指定等待執行程式單元之優先順序。

六、 有一個機制可讓某一個並行程式單元延遲一段有限時間。

可就此六項功能討論C語言之能力。

(二)

Monitor平行語言之優點：

1. 保證在某一時間內只有一個處理程序能執行monitor中的程序。
2. monitor中的程序只能存取區域性資料和實際參數，而外界也不可以存取monitor中的區域性資料和區域性程序。
3. 當某一個處理程序正在monitor中執行時，外面可能有玲個、一個或多個處理程序在queue中等待進入此monitor。

與semaphore比較下：

1. semaphore在每一個處理程序都保有自己的臨界區域。
2. monitor將所有用到的共享資料及相關運算都收集在一個語法單元(即monitor)內；可完成抽象資料型態。

Message passing平行語言之優點：

1. 以工作為並行單元。
2. 當Task A傳送訊息給Task B，而Task B正在忙的時候，Task B的程序不會被中斷，只有當Task B完成工作時才會去處理Task A送來的訊息。
3. 一個Task可以擱置自己的執行，以等待別的Task傳送訊息過來。
4. 當Task A要傳送訊息給Task B，而Task B正在等待接受訊息，則這個訊息可以被傳送，此時真正把訊息傳遞過去執行，稱之為『集結』或『會晤』。
5. 當Task B正在處理Task A傳送過來的訊息時，任何從其他Task傳送過來的訊息都將被儲存在Task B的Queue中，當Task B完成了目前的工作，再從Queue中取出一個訊息來加以處理。

與semaphore比較下：

1. 使用semaphore時，當wait誤寫成release，則可能讓兩個處理程序同在臨界區域中。
2. 若release誤寫成wait，則可能發生『死結』現象。