《程式語言》

一、程式在計算機裡執行的時候，必須妥善安排變數的放置區塊。不同種類的變數應安置於不同的區塊，以便管理，增進程式執行的效率。下列程式裡的七個變數 a, b, c, d, e, f, *f 各應置於何處？（20 分）

```
#include <stdlib.h >
int a;
static int b;
int foo (int c) {
   int d, *f;
   static int e;
   b = 2 * c;
   d = a + b + c;
   e = d * 3;
   f = (int *)malloc(sizeof(int));
   *f = e + 7;
   return e * 5;
}
int main(){
   a = 5;
   b = 4;
   a = foo(b);
}
```

命題意旨	本題在測驗考生對於記憶體繫結之概念掌握度，包括四種記憶體繫結其發生原因及所儲存在記憶體中對應區段，相似於 105 年關務 4 等、104 年鐵路 3 等及 102 年檢事官之考點。
答題關鍵	需以表格或條列式說明各種變數之存放區塊。
考點命中	《高點程式語言講義》第三回，金乃傑編撰，儲存類別與生命期部分，頁 73-74。

【擬答】

以下依照題意將各變數之記憶體儲存區塊以表格整理之：

變數	類型	存放區塊
a	全域變數	Data Segment 中的 BSS 區段
b	靜態變數	Data Segment 中的 BSS 區段
c	參數	Stack Segment 中的活動紀錄（Activation Record）參數區域
d	區域變數	Stack Segment 中的活動紀錄（Activation Record）區域變數區域
e	靜態變數	Data Segment 中的 BSS 區段
f	區域變數	Stack Segment 中的活動紀錄（Activation Record）區域變數區域
*f	外顯堆積動態變數	Data Segment 中的 Heap 區段

二、請用下面的「與前後文無關的文法」（context-free grammar）為下面的程式畫出程式結構樹（concrete syntax tree）。（20 分）

program var a, b : int; a := 0; b:= a + 1; end.

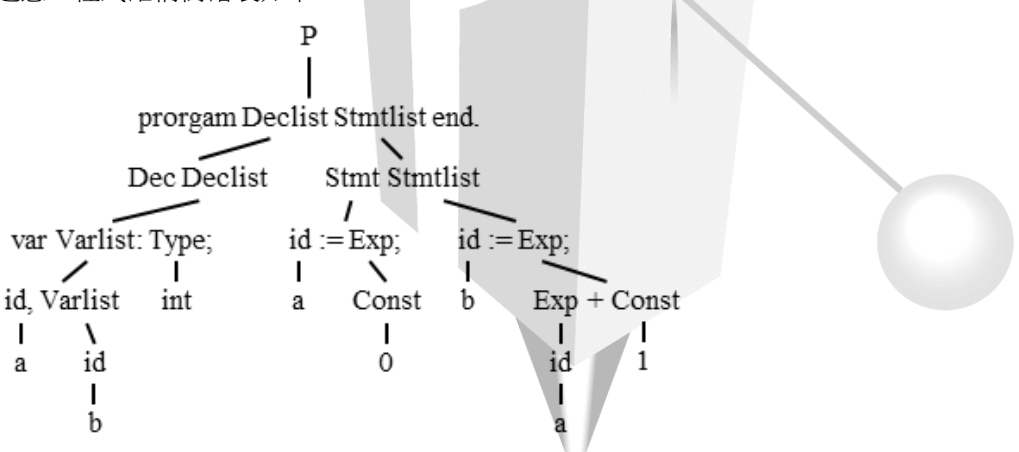
```
P ::= program Declist Stmtlist end.
Declist ::= Dec Declist
Declist ::=
Dec ::= var Varlist : Type;
Varlist ::= id
```

Varlist ::= id, Varlist
Type ::= real
Type ::= int
Stmtlist ::= Stmt Stmtlist
Stmtlist ::=
Stmt ::= id := Exp;
Exp ::= Const
Exp ::= Exp + Const
Exp ::= id
Const ::= 0
Const ::= 1

命題意旨	本題在測驗考生對於剖析樹之繪製，及上下文無關文法的了解程度，能根據 BNF 文法將語句轉換為剖析樹。相似於 104 高考及 104 地方特考 3 級考試之考點。
答題關鍵	根據題意繪製剖析樹。
考點命中	《高點程式語言講義》第一回，金乃傑編撰，BNF 文法與剖析樹部分，頁 8-9。

【擬答】

依照題意，程式結構樹繪製如下：



註：Concrete syntax tree 是具體樹，又稱 Parse Tree（剖析樹）。

三、幾乎所有的程式都需要一些支援程式，例如輸出輸入、數學函式、記憶體管理等。這些支援程式一般都是放在支援程式庫（libraries）裡面，然後在適當的時機和主程式連結（linking）。連結的時機可以分為兩大類：靜態連結（static linking）及動態連結（dynamic linking）。請舉例解釋靜態連結及動態連結，並請說明產生與使用相關程式庫的方法。（20 分）

命題意旨	本題在測驗考生對於編譯器之靜態連結與動態連結相關之知識，必須了解產生及使用的相關方法。
答題關鍵	本題解題層次為： 1.說明靜態連結之定義（優缺點）、產生及使用方法。 2.說明動態連結之定義（優缺點）、產生及使用方法。
考點命中	《高點程式語言講義》第六回，金乃傑編撰，編譯器詳細流程部分，頁 53-54。

【擬答】

(一)靜態連結（Static Linking），指程式在編譯期間由編譯器（Compiler）與連結器（Linker）將靜態函式庫整合至應用程式內，並製作成目的檔以及可以獨立運作的執行檔。靜態連結之主要優點是程式和程式相關函式庫都保存在一起，因此只需保證在編譯時期有正確引入函式庫檔案，程式發布時不用擔心使用者電腦中是否有相關函式庫或函式庫之版本，都可以正常執行；缺點是二進位執行檔體積較大，而且如果函式庫有更新的話，整個執行檔也要跟著重新編譯。以下說明在 gcc 中產生及使用的方法：

1.靜態函式庫可以是一般的 C 語言檔案，如 svm.c，在 gcc 中可以如下編譯：
gcc -c -o svm.o svm.c
2.靜態函式庫通常為 lib 開頭，.a 結尾的檔案，需透過 ar 指令產生：
ar -rcs libsvm.a svm.o
3.在需要用到該靜態函式庫的 main.c 檔案中，可以使用如下指令連結以產生 main_static 執行檔：
gcc main.c libsvm.a -o main_static

(二)動態連結（Dynamic Linking）的函式會在程式執行時才被載入，而不是直接編譯在執行檔中，這種作法讓系統更彈性的應用硬體資源，而且未來共享函式庫在更新之後，執行檔也不需要重新編譯，此外還可以不公開程式碼的情況分享給別人使用。值得一提的是，在 gcc 中還可分為共享函式庫（Shared Library）與動態載入函式庫（Dynamically Loaded Library）兩種，共享函式庫在程式載入時就會全部載入；而動態載入函式庫如 Windows 中的.dll 檔案，在執行到有需要時才會載入。以下說明在 gcc 中產生及使用動態載入函式庫的方法：

1.動態載入函式庫因為在執行時程式的位置才被確認，因此在編譯時需要使用-fPIC 指令，產生位置獨立的程式碼（Position Independant Code, PIC）：
gcc -c -fPIC -o svm.o svm.c
2.動態函式庫通常為 lib 開頭，.so 結尾（在 Linux 中），使用-Wl 參數設定參數給連結器，將 svm.o 建立為共享函式庫，其中 libsum.so.1.0.0 代表介面版本為 1，0 個新增介面，0 個修改；如果介面增加則中間的數字（mirror）會增加，如果介面修改則版本號就要增加：
gcc -shared -Wl,-soname,libsvm.so.1 -o libsvm.so.1.0.0 svm.o
3.當建立共享函式庫後，要建立不含版本號的連結檔案，讓程式指向：
ln -s libsvm.so.1.0.0 libsvm.so
4.在撰寫程式時，若需要用到動態載入函式庫，程式中會有如下語法：
handle = dlopen ("libsvm.so.1", RTLD_LAZY);
5.在編譯開程式時，可以使用如下指令連結以產生 main_dl 執行檔：
gcc main_dl.c -ldl -o main_dl
6.最後在執行時需要告知程式動態載入函式庫的路徑，以讓函式庫可以正常執行：
LD_LIBRARY_PATH=. ./main_dl

四、請問下面的 C++程式經由編譯器編譯時，是否有錯誤發生？若有錯誤，錯誤為何？若無錯誤，執行此程式印出的結果為何？（10 分）

```
#include<stdio.h >

class A{
  protected:
    int X;
  public:
    A(){X = 1;}
};
class B : public A{
  private:
    int X;
  public:
    B(){X = 200;}
};
class C : public B{
  public:
    void print C() { printf("X is %d.\n",X);}
};
```


命題意旨	本題在測驗考生對於物件導向程式語言中資訊隱藏常使用之保護層級保護功能之掌握，須對繼承關係中 public、protected 及 private 了解。

答題關鍵	需指出程式中之錯誤，並說明錯誤原因。
考點命中	《高點程式語言講義》第四回，金乃傑編撰，封裝與資訊隱藏部分。

【擬答】

該程式有錯誤發生，因為類別 C 中無法存取類別 B 中設為 private 的整數變數 X。

五、Continuation 是函數式程式語言的程式技巧。下面這個 Scheme 程式展示了 continuation 的使用方法。請問執行此程式的結果為何？並請說明計算的過程。（20 分）

```
(define qq (lambda (n cont)
      (map (lambda (x) (if (negative? x) (cont (* -1 x)) x))
      '(54 0 37 -30 -28 83))))

( + 100 (call-with-current-continuation
   (lambda (exit) (qq 5 exit))))
```

命題意旨	本題在測驗考生關於 Scheme 語言的 lamdba、map 及 call-with-current-continuation 等函數運作方式的了解。
答題關鍵	解題層次為： 1.先寫出程式執行結果 2.再說明出現此結果之原因
考點命中	《高點程式語言講義》第五回，金乃傑編撰，函數型程式語言部分。

【擬答】

依照題意程式輸出如下：

130

Continuation 只的是「值相關運算」，如程式碼第二行，if 會根據 x 是正負數來決定執行(cont (* -1 x))還是 x；而 call-with-current-continuation 是一個特殊語法，可用以取出在 if 成立時繫結到參數的值，用以觀察目前「值相關運算」的處理結果。

qq 函數有 2 參數：n 及 cont，n 為一整數，cont 為一函數。在 qq 執行時，會使用 map 將串列(54 0 37 -30 -28 83)中的每個元素套用到第二行的匿名函數，再透過 if 敘述判斷該套用的值。若串列中傳入的數為負數，就將之變為正，並將之變為傳入 cont 函數的參數，將函數回傳結果放入新串列中；否則直接將串列中的數字放入新串列中。值得一提的是，在 qq 中，參數 n 是沒有用的，不管帶入什麼都不會影響結果。

考慮執行以下指令，透過 lamdba 生成一個匿名函數給 cont 帶入：(qq 5 (lambda (n) (+ n 1000)))

執行該指令，會回傳(54 0 37 1030 1028 83)串列，該串列將-30 及-28 變為正數並加 1000。

而題目中以 call-with-current-continuation 取出第一次讓 if 為 true 的數，串列中的-30；根據程式語法，此時會將-30 轉為正數回傳，傳回之 30 與外層 100 相加，故得到 130。

六、現今網路愈來愈發達，人們常由手機或是平板電腦造訪網頁，並且下載應用程式（Application, APP）來執行程式。這樣做會遇上兩項重大的問題。請問此兩大問題為何？如何由電腦系統來解決或是預防此二問題？（回答程式語言與系統方面相關的問題，請勿回答網路速度、基地台、通信協定等網路與通信的因素。）（10 分）

命題意旨	本題在測驗考生對於行動時代網路環境的了解，包括行動網站、行動應用程式及其平台。
答題關鍵	解題層次為： 1.先指出兩大問題 2.針對兩大問題分別撰寫形成原因及解決方法
考點命中	《高點程式語言講義》第五回，金乃傑編撰，網路應用程式部分。

【擬答】

在行動裝置瀏覽網頁及下載行動應用程式遇到最大的問題為跨平台顯示及應用程式相容性，以下分別說明其內容與解決方法：

(一)跨平台顯示：

1.以往網頁都是設計給電腦瀏覽，因電腦螢幕屬於橫向螢幕，因此常有兩欄或三欄設計，在左右邊的欄中加入導覽或分類選項。但因為行動裝置瀏覽多以直向螢幕，螢幕大小又不如電腦，因此在行動裝置上常會遇到畫面被壓縮在極小的比例上，文字必須要手動放大才能閱讀，也難以在網頁中找尋相關資訊。

2.解決方法：使用響應式網頁設計（Responsive web design, RWD），RWD 設計主要透過 HTML 與 CSS 語法，透過 HTML 的 meta 標籤設定 viewport 的縮放比例（在手機上讓網頁寬度調整成螢幕寬度），並搭配@media 這種特殊 CSS 語法針對不同的螢幕解析度分別設置在特有解析度所需呈現的 CSS，例如當設置 @media (max-width:770px)代表當螢幕寬度最寬是 770px 時要額外套用的 CSS 語法，以針對行動瀏覽器的呈現方式優化。

(二)應用程式相容性：

1.因目前行動平台分屬於 Android 與 iOS 兩大陣營，而此兩大陣營相對應的應用程式不同，因此開發者在開發應用程式時通常都必須針對不同平台各自開發，並力求兩程式一致，增加許多開發成本與難度；此外，對於應用程式支援的功能也會因平台而有不同，例如蘋果 iOS11 之後支援擴增實境功能，而 Android 卻未必能支援，或某些應用程式需要 NFC 感應功能，但行動裝置未必提供；最後，也因為兩大陣營擁有各自的平台，因此下載網址也不相同，造成使用者搜尋安裝的困擾。

2.解決方法：若應用程式之功能對裝置依賴度低，如不涉及推播、背景執行、硬體相關功能，可選擇以 HTML5 開發 Web App，即可輕易達成跨平台應用，另外亦可選擇開發 Chatbot，整合至 Facebook Messenger 或 LINE，讓使用者可在任何平台上透過即時通訊互動；若對裝置依賴度高，亦可將基礎功能以 Web App 開發，在使用者欲使用進階功能時再以網頁語言偵測使用者所用之平台，提供對應之下載連結。