1. 閱讀完記憶體布局的方式後，請推論並回答下列問題：

   (a) 下列敘述何者為真？

   (A) 記憶體布局存在標準，所有作業系統都必須遵循標準的布局系統
   (B) 在同一個作業系統上，不同的 C 語言程式可能會有不同的記憶體布局
   (C) 有時候遇到遞迴過深的問題可以使用修改程式以外的方式解決
   (D) 如果遞迴太深，會把 heap 的記憶體用光，導致 runtime error

   解答

   (D) 應該是把 stack 的記憶體用光

   正確選項：(C)

   (b) 關於各個記憶體區塊的特性，下列何者為非？

   (A) 一樣的 code 在不同編譯器下編譯後形成的執行檔大小不一定一樣，因為不同編譯器轉換成機器碼的方式可能不同
   (B) 靜態變數和區域變數如果沒有指定初始值，預設有可能會是任意的值
   (C) 因為編程者指定的初始值會存在 initialized data segment，所以手動初始化的變數越多，一般而言編譯後的執行檔會越大
   (D) 在程式開始執行後，位於 initialized data segment 和 uninitialized data segment 的資料都還是有可能會被修改

   解答

   (B) 靜態變數如果沒有指定初始值，會位於 uninitialized data segment，該區記憶體會自動初始化為 0

   正確選項：(B)

   (c) 下列基於上述記憶體布局的推論，何者正確？

   (A) 如果我們同時運行兩個相同的程式 (但不一定做相同的操作，如同時開兩個 skype 視窗跟不同的人聊天)，那麼我們可以透過兩者共享相同的 initialized segment data 記憶體來節省空間
   (B) 如果我們同時運行兩個相同的程式 (但不一定做相同的操作，如同時開兩個 skype 視窗跟不同的人聊天)，那麼我們可以透過兩者共享相同的 text segment 記憶體來節省空間
   (C) 假如某個程式中有兩個 function $main(), f()$，並且執行順序為：
      1. 執行 $main()$
      2. 呼叫並執行 $f()$
      3. 退出 $f()$，回到 $main()$

4. 退出 $main()$，程式結束

那麼在執行期間，任意 $f()$ 內的變數的記憶體位置一定都比任意 $main()$ 內的變數的記憶體位置還要小

(D) 在 Linux 上我們可以透過修改編譯參數來增加 stack 深度的上限

解答

(A) 因為做不同操作，所以開始執行後兩個程式的 initialized segment data 可能會不同，屆時便無法共享

(B) 正確解答

(C) 在不同作業系統底下，stack 不一定都是往記憶體位置較小的方向成長，且函數中也可能有靜態變數

(D) Windows 上才可以

正確選項：(B)

2. 有份程式碼如下：

```
1  int var1;
2  static int var2;
3  int var3 = 3;
4
5  int f(int var4){
6      if( var4 <= 1 ) return 1;
7      else return f(var4-1) + f(var4-2);
8  }
9
10 int main(){
11     int var5;
12     static int var6;
13     int *var7 = (int*)malloc(sizeof(int)*100);
14     int var8 = 5;
15     return 0;
16 }
```

請問程式碼中，各個變數分別儲存於記憶體分佈中的哪個區塊呢？請使用代號 (A~E) 回答問題。

選項：

(A) Text Segment

(B) Initialized Data Segment

(C) Uninitialized Data Segment

(D) Heap

(E) Stack

(a) var1

(b) var2

(c) var3

(d) var4

(e) var5

(f) var6

(g) var7

(h) var8

(i) *var7

(j) main

(k) main()

解答

(a) 未初始化的全域變數

(b) 未初始化的靜態變數兼全域變數（兩者都該是在 uninitialized data segment）

(c) 有初始化的全域變數

(d) 區域變數

(e) 區域變數

(f) 未初始化的靜態變數

(g) var7 是一個型態為 int*（指向 int 型態的指標）的區域變數

(h) 區域變數

(i) *var7 的儲存空間是以 malloc 取得的，位於 heap 中

(j) main 是一個函數，標記於執行檔中一串指令的開頭，因此在 text segment

正確選項：(C)(C)(B)(E)(E)(C)(E)(E)(D)(A)