《程式語言》

一、(一)請將下面程式改寫為尾遞迴（tail recursion）的形式。（15分）

```
int recsum(int x){
   if (x == 1) return(x);
   return(x + recsum(x - 1));
}
```

(二)請問尾遞迴形式的優點為何？（10分）

試題評析	本題為104年身心障礙程式語言考試之相似題，主軸為「尾端遞迴」。由於過去10年國家考試申論中只有一題尾端遞迴的題目，因此考生在準備時容易忽略。 但若考生能掌握尾端遞迴概念，確保return中沒有其他的運算，其實題目範圍並沒有太大的變動。只要將課本尾端遞迴範例的「階層」改為題目中的「連加」即可輕易回答，屬於有準備就有分數的題目。
考點命中	1.《高點・高上程式語言總複習講義》，金乃傑編撰，考點5。 2.《高點・高上程式語言講義》第二回，金乃傑編撰，頁48。

答：

(一)依照題意將程式改寫如下：

```
6  int recsum_tail(int x, int sum = 1){
7      if (x == 1) return sum;
8      else return recsum_tail(x - 1, sum + x);
9  }
```

(二)尾遞迴之優點如下：

1.執行效率更佳：因為遞迴函式所使用的活動紀錄格式相同，而尾遞迴在回傳時沒有其他運算，因此可以直接將回傳的內容帶入上一層函式中，如上例呼叫recsum_tail(5)時，可以直接將return代換為：1 + 5 + 4 + 3 + 2，編譯器的最佳化程式可以省去為函式呼叫建立活動紀錄，只需要做代換即可得到結果。

2.程式可讀性更高：相較於迴圈，遞迴程式的可讀性更高，便於撰寫、維護與了解邏輯；另一方面，由於遞迴程式的結構，也可以使用到較少區域變數，使程式看起來更簡潔。

二、假設每個int變數占用4 bytes，每個指標變數也占用4 bytes。下面的C程式印出的結果為何？（25分）

```
# include <stdio.h>
typedef int T1[10][9];

int main(){

   struct {
      T1 *a[10];
      int (*b)[100];
   } f[10][10][10];

   printf("p1 = %d\n",(int)sizeof(f[1][5]));
   printf("p2 = %d\n",(int)sizeof(f[2][3][4].a));
   printf("p3 = %d\n",(int)sizeof(f[3][2][6].b));
   printf("p4 = %d\n",(int)(f - &f[5]));
   printf("p5 = %d\n",(int)(f[6][2] - f[3][3]));
}
```

試題評析	本題為105年高考、106年地特四等程式語言計算陣列記憶體空間的相似題，屬於考生必須要熟稔掌握之題型。以題目結構而言，內容沒有太大變化，但是本題結合結構(struct)與指標概念，使作答難度變高。 若考生能熟悉結構的空間表示方式，且能清楚陣列指標與指向一個陣列的指標寫法，就可以快速判別所佔的空間，本題應可拿到高分。
考點命中	1.《高點・高上程式語言總複習講義》，金乃傑編撰，考點3。 2.《高點・高上109高普考題神》程式語言，金乃傑編撰，考點1。 3.《高點・高上程式語言講義》第三回，金乃傑編撰，頁16-17。

答：
該程式輸出如下：
p1 = 440
p2 = 40
p3 = 4
p4 = -5
p5 = 290

說明：
T1所佔空間為：360 Bytes (10 * 9 * 4)
f的每個元素佔：44 Bytes (10 * 4 + 4)，a佔40；b佔4
故f[1][5]共10個元素佔440 Bytes

三、下面的C程式印出的結果為何？（作答必須解釋計算過程，只寫答案而未加解釋，只能得部分分數。）（25分）

```
# include <stdio.h>

int foo1(int p){
   if (p >= 90) return(foo1(foo1(p-11)));
   return(p+10);
}
int foo2(int p){
   if (p < 91) return(foo2(p+11));
   return(p);
}

int foo(int p){
   return(foo1(foo2(p)));
}

int main(int argc, char **argv){
   int q;
   q = 65;
   printf("foo(%d)= %d.\n", q, foo(q));
   q = 83;
   printf("foo(%d)= %d.\n", q, foo(q));
   q = 95;
   printf("foo(%d)= %d.\n", q, foo(q));
   q = 100;
   printf("foo(%d)= %d.\n", q, foo(q));
   q = 142;
```

```
    printf("foo(%d)= %d.\n", q, foo(q));
    return(0);
}
```

試題評析	本題亦為遞迴題，相較於往年撰寫遞迴程式之題型，本題僅需要追蹤程式碼即可。不過值得注意的是，本題的每小題跑出來的答案均相同，故題目要求須加以說明。 本題利用特定的遞迴規則使foo1()與foo2()在一定的情況下會有同樣的輸出，再配合固定的順序：先呼叫foo2()再呼叫foo1()，使不論傳入什麼參數都有相同的輸出，可以看見巧思，但可能出在高考中尚缺乏鑑別度。 考生若能掌握其邏輯規則，不要掉入題目陷阱，要得滿分並非難事。
考點命中	1.《高點・高上程式語言總複習講義》，金乃傑編撰，考點5。 2.《高點・高上109高普考題神》程式語言，金乃傑編撰，考點3。 3.《高點・高上程式語言講義》第二回，金乃傑編撰，頁30-32。

答：

該程式輸出如下：

foo(65)= 99.
foo(83)= 99.
foo(95)= 99.
foo(100)= 99.
foo(142)= 99.

說明：

1.foo1若發現參數小於90，會直接將該參數+10並輸出；反之會將該參數減11，並遞迴呼叫自己。故若foo1接到之參數大於等於90，經過foo1調整，最後參數都會變為89，並輸出99。
2.foo2若發現參數大於等於91，會直接將參數輸出；反之則會將該參數加11後，並遞迴呼叫自己。
3.由於foo2在foo1之前完成，故透過foo1處理之回傳值必大於等於91；又由於foo1只要參數大於等於90就會輸出99，故該程式總是輸出99。

foo(65)
= foo1(foo2(65)) = foo1(foo2(76)) = foo1(foo2(87)) = foo1(foo2(98))
= foo1(98) = foo1(foo1(87)) = foo1(97) = foo1(foo1(86)) = foo1(96) = foo1(fool(85)) = foo1(95) … foo1(89) = 99
foo(83)
= foo1(foo2(83)) = foo1(foo2(94))
= foo1(94) = foo1(foo1(83)) = foo1(93) = foo1(foo1(82)) = foo1(92) = foo1(fool(81)) = foo1(91) … foo1(89) = 99
foo(95)
= foo1(foo2(95))
= foo1(95) = foo1(foo1(84)) = foo1(94) = foo1(foo1(83)) = foo1(93) = foo1(fool(82)) = foo1(92) … foo1(89) = 99
foo(100)
= foo1(foo2(100))
= foo1(100) = foo1(foo1(89)) = foo1(99) = foo1(foo1(88)) = foo1(98) = foo1(fool(87)) = foo1(97) … foo1(89) = 99
foo(142)
= foo1(foo2(142))
= foo1(142) = foo1(foo1(131)) = foo1(141) = foo1(foo1(130)) = foo1(140) = foo1(fool(129)) = foo1(139) … foo1(89) = 99

四、物件導向程式語言有繼承的觀念，請解釋單一繼承（single inheritance）與多重繼承（multiple inheritance）的意義、差別及實作方法。（25分）

試題評析	本題相似於104年警察、104年身心障礙的題目，尤其104年警察就已經考過多重繼承與單一繼承之優缺點，即可作為「差異」的比較內容，故對於已掌握考古題的考生而言，作答此題並非難事。惟該題配有25分，1/4張考卷的分數，應對「差別」有更完整的解釋，並使用程式碼演示兩種既成的實際方式，方能得到高分。

考點命中	1.《高點・高上程式語言總複習講義》，金乃傑編撰，考點14。 2.《高點・高上109高普考題神》程式語言，金乃傑編撰，考點5。 3.《高點・高上程式語言講義》第四回，金乃傑編撰，頁132-133、192-193。

答：

在物件導向程式語言中，繼承(Inheritance)提供子類別使用父類別的方法跟屬性，也提供overriding，將父類別的方法重新定義，提供較符合子類別中所需的版本。以下以C語言舉例，比較兩者之意義、差別與實作方法：

	單一繼承single inheritance	**多重繼承multiple inheritance**
意義	子類別中的方法或屬性只能繼承自「一個」父類別。	子類別可以同時繼承「多個」父類別的方法或屬性。
優點	1.可讀性高：一個類別只會對應到一個父類別，結構單純，使程式容易維護也容易除錯。 2.實作簡單：在實作時不需要考慮到多個父類別的情形，因此儲存類別資料的空間只需要依照為一個父類別進行設計，在實作多型時也可以輕易讓父類別的參考指向子類別的物件。 3.可避免「鑽石問題」：因為只有唯一的繼承鏈，不必擔心重複繼承問題。	1.意義性更接近真實世界：舉例而言，機器人可能就同時具備「人」與「機器」的特性，人會思考機器會工作，此種狀態在多重繼承中可以輕易表達，使用一類別同時繼承人與機器即可。 2.彈性高：不同類別功能不同，若需要同時具備兩個以上的功能，而原先類別又不需要繼承關係，使用多重繼承可以輕易達成，增加程式彈性。
缺點	1.無法完全描述真實世界需求：在真實世界中可能繼承自多種類別的屬性與方法，但若只有單一繼承無法表示此現象。 2.彈性較低：若需要同時繼承兩個類別的屬性或方法，則此兩類別必須為繼承關係，再繼承才能有此效果，這樣未必符合需求，也缺乏彈性。	1. 具備「鑽石型繼承樹(Diamond Of Death, DoD)」問題：此問題來自於多個父類別的父類別相同，使子類別使用到祖父類別的屬性或方法時，因為可能在父類別中被修改過，則可能產生模糊的問題。 2.建立多型困難：多型建立在使用父類別的參考操作子類別物件。若使用多重繼承，則使用哪一個父類別的參考可能都會遇到問題，因此可能需要使用dynamic_cast()進行轉型後才能使用，如此也會喪失執行效率。
適用性	程式結構簡單，表示問題單純且多人開發環境。	文件詳細，程式需要展現複雜問題的邏輯。

實作方法	class Parent{
	}
	
	class Child: public Parent{
	//單一繼承Parent
	}
		class Father{
		}
		
		class Mother{
		}
		
		class Child: public Father, public Mother{
		//多重繼承Father及Mother
		}