109 年公務人員高等考試三級考試試題

類　科：資訊處理
科　目：程式語言

一、㈠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));
}
```

㈡請問尾遞迴形式的優點為何？(10 分)

《考題難易》：★★
《破題關鍵》：本題為遞迴經典考古題，掌握尾端遞迴方式即可得到解答。

【擬答】

㈠上列程式可修改成下列尾端遞迴

```
int tailrecsum(x, running_total=0){
  if (x == 0)
    return running_total
  else
    return tailrecsum(x - 1, running_total + x)
}
```

㈡若函式在尾位置呼叫自身(或是一個尾呼叫本身的其他函式等等),則稱這種情況為尾遞迴。尾遞迴也是遞迴的一種特殊情形，可通過最佳化，使得計算僅占用常數堆疊空間，也比較容易實現。例如㈠原題呼叫 recsum(5)過程如下：

```
5 + recsum(4)
5 + (4 + recsum(3))
5 + (4 + (3 + recsum(2)))
5 + (4 + (3 + (2 + recsum(1))))
5 + (4 + (3 + (2 + 1)))
5 + (4 + (3 + 3))
5 + (4 + 6)
5 + 10
15
```

可以發現堆疊占用了大量空間

如㈠改用尾遞迴後，改為呼叫 tailrecsum(5, 0)過程成為

```
tailrecsum(5, 0)
tailrecsum(4, 5)
tailrecsum(3, 9)
tailrecsum(2, 12)
tailrecsum(1, 14)
tailrecsum(0, 15)
15
```

此時對記憶體的消耗是線性的，可以大幅降低資源需求。

二、假設每個 int 變數占用 4byles，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]));
}
```

《考題難易》：★★★
《破題關鍵》：掌握 C 語言矩陣與結構處理方式即可得到解答。

【擬答】

p1 = 880
p2 = 80
p3 = 8
p4 = -5
p5 = 290

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 foo0(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);
}
```

《考題難易》：★★★

《破題關鍵》：本題為遞迴運算實作題，需要清楚且耐心的逐步推論才能得到正確解答，但今年普考程式設計概要考出幾乎完全相同題目，不過推導過程較為繁複，但仍可在幾次推導後找出規律。

【擬答】整體執行過程如下：

㈠由 main 作為程式進入點開始執行，先設定 q=65，因此呼叫

foo(65)
=foo1(foo2(65))
=foo1(foo2(76))
=foo1(foo2(87))
=foo1(foo2(98))
=foo1(98) 由此處開始，此一遞迴呼叫有固定模式
=foo1(foo1(87))
=foo1(97))
=foo1(foo1(86))
=foo1(96))
=foo1(foo1(85))
=foo1(95))
=foo1(foo1(84))
=foo1(94))
=foo1(foo1(83))
=foo1(93))
=foo1(foo1(82))
=foo1(92))
=foo1(foo1(81))
=foo1(91))
=foo1(foo1(80))
=foo1(90))
=foo1(foo1(79))
=foo1(89))
=99 因此

foo(65)= 99. 可以發現呼叫 foo1 時，最後會收斂在 99

㈡設定 q=83，因此呼叫

foo(83)

=foo1(foo2(83))

=foo1(foo2(94))

=foo1(94) 此處與㈠推導中呼叫 foo1(94)相同，可推論後面推導過程相同，故省略

=...

=99 因此

foo(83)= 99.

㈢設定 q=95，因此呼叫

foo(95)

=foo1(foo2(95))

=foo1(95) 此處與㈠推導中呼叫 foo1(95)相同，可推論後面推導過程相同，故省略

=...

=99 因此

foo(95)= 99.

㈣設定 q=100，因此呼叫

foo(100)

=foo1(foo2(100))

=foo1(100)

=foo1(foo1(89))

=foo1(99)

=foo1(foo1(88))

=foo1(98) 此處與㈠推導中呼叫 foo1(95)相同，可推論後面推導過程相同，故省略

=...

=99 因此

foo(100)= 99.

㈤設定 q=142，因此呼叫

foo(142)

=foo1(foo2(142))

=foo1(142)

=foo1(foo1(131))

=foo1(foo1(foo1(120)))

=foo1(foo1(foo1(foo1(109))))

=foo1(foo1(foo1(foo1(foo1(98)))))

=foo1(foo1(foo1(foo1(foo1(foo1(87))))))

=...根據前面的推導，foo1 只會到 p=99 才會穩定

= 99 因此

foo(142)= 99.

因此本題完整輸出如下：

foo(65)= 99.

foo(83)= 99.

foo(95)= 99.

foo(100)= 99.

foo(142)= 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)

《考題難易》：★★

《破題關鍵》：了解多重繼承意義，並掌握 C++與 Java 的多重繼承實作方式即可得到解答。

【擬答】

㈠物件導向程式設計中的多重繼承（multiple inheritance）指的是一個類別可以同時從多於一個父類繼承行為與特徵的功能。與單一繼承相對，單一繼承指一個類別只可以繼承自一個父類。

㈡單一繼承與多重維承最大的差別就是所繼承的類別數量，單一繼承時子類別繼承了父類別的特徵，允許分享功能。例如，可以創造一個「哺乳類動物」類別，擁有進食、繁殖等的功能；然後定義一個子類型「貓」，它可以從父類別繼承上述功能，不需重新編寫程式，同時增加屬於自己的新功能，例如「追趕老鼠」。但是多重繼承可以同時自多於一個結構繼承，例如容許「貓」繼承「哺乳類動物」之餘，同時繼承「卡通角色」和「寵物」，缺乏多重繼承往往會導致十分笨拙的混合繼承，或迫使同一個功能在多於一個地方被重寫。（這帶來了維護上的問題）

㈢Java 允許一個類別繼承自多於一個父介面（可以指定某一個類別，它繼承了所有父類別的類型，並必須擁有所有父類別介面的外部可見方法的具體實現，並允許編譯器強制以上要求），但只可以從一個父類別繼承實現（方法與資料）。C++支援多重繼承，允許對現實世界進行更直接的建模，Borland C++的 OWL Framework 大量使用多重繼承來描述視窗的關係。