臺北市一Ｏ二學年度高級中等學校電腦程式設計競賽決賽試題

(高中組)

說明：

1. 本試卷共有四題，每題 25 分。
2. 請記得隨時備份自己的程式。

試題 1：二元加法

問題敘述

將 n (n>=3)個正整數加總起來是一件很簡單的事，但現在我們僅能運用 2 元加法來完成它，所謂 2 元加法就是每次僅能有兩個數相加，且每次加完的和會成為下次加法的被加數，並無額外暫存器，而每次相加的和就是所付出的成本(cost)。我們發現運用 2 元加法的計算過程(順序)不同就會產生整個計算成本(cost)不同的問題。請設計一程式運用二元加法將 n 個正整數相加，並找出最佳的相加順序，使其所付出的成本最低。

例：將 1,2,3 相加可以有下列三種可能，其中(A)的總成本最低。

(A) 1 + 2 = 3, cost1 = 3; 3 + 3 = 6, cost 2= 6; Total cost = 9

(B) 1 + 3 = 4, cost1 = 4; 4 + 2 = 6, cost2 = 6; Total cost = 10

(C) 2 + 3 = 5, cost1 = 5; 5 + 1 = 6, cost2 = 6; Total cost= 11

輸入說明

讀入一個檔案(檔名為 in.txt)，內有數組測試資料，每組有兩列資料，第一列為一正整數 n，第二列為 n 個數值(正整數)，當 n=0 時表示輸入結束。

輸入檔範例

```
5
16, 21, 9, 30, 9
4
10, 3, 20, 7
0
```

輸出說明

請將輸出資料顯示在螢幕上，顯示加總之總和與加總所需付出的最小成本。

輸出範例

```
85,192
```

40, 70

試題 2. 登山演算法 (本題無輸入資料，只需輸出答案，Output only)

問題敘述

登山演算法 (hill climbing) 是一種簡單的近似最佳化演算法。以登山演算法求解問題的步驟如下：

(1) 待求解問題為 Maximize $F(s)$，$s \in \Omega$。$F(.)$ 表示目標函式，Ω 表示解空間，s 表示一個解。目標為求取最佳解 s^* 使得 $\forall\, s \in \Omega$，$F(s^*) \geq F(s)$。
(2) 產生問題的初始解 s_0（通常會以隨機方式或者以領域知識來產生），令基底解 $s = s_0$。
(3) 使用鄰域函式 (neighborhood function) N 來產生 s 鄰近的解，令 $N(s)$ 代表 s 鄰近解的集合。
(4) 找出 $N(s)$ 中最佳的解 $s' = \arg\max\{F(i) \mid i \in N(s)\}$。
(5) 如果 $F(s') > F(s)$，$s = s'$，回到步驟 (3)；否則進入步驟 (6)。
(6) 輸出 s 為登山演算法找到的近似最佳解。

登山演算法常被用來解決 NP-hard 最佳化問題，但在這道試題中，我們將用它來解決一個簡單的函式最大化問題。假設有個函式 $F(x)$ 定義如下：

$F(0) = 325$	$F(1) = 56$	$F(2) = 1188$	$F(3) = 99$
$F(4) = 13$	$F(5) = 600$	$F(6) = 9999$	$F(7) = 100$

我們希望以登高演算法找出 $0 \leq x \leq 7$ 中，能最大化 $F(x)$ 的 x 值。此處我們以二進位方式來表示每個解，並且以翻轉 (flip) 一個位元作為鄰域函式。如果以 $x = (000)_2$ 為初始值，則登高演算法的求解過程如下：

$s = (000)_2$ $F((000)_2)=325$ $F((001)_2)=56$ $\boldsymbol{F((010)_2)=1188}$ $F((100)_2)=13$	$s = (010)_2$ $F((010)_2)=1188$ $F((011)_2)=99$ $F((000)_2)=325$ $\boldsymbol{F((110)_2)=9999}$	$s = (110)_2$ $\boldsymbol{F((110)_2)=9999}$ $F((111)_2)=100$ $F((100)_2)=13$ $F((010)_2)=1188$
以 $(000)_2$ 為基底解，翻轉每個位元以產生三個鄰近解。鄰近解中的最佳解 $(010)_2$ 比 $(000)_2$ 好，故取代之。	以 $(010)_2$ 為基底解，翻轉每個位元以產生三個鄰近解。鄰近解中的最佳解 $(110)_2$ 比 $(010)_2$ 好，故取代之。	以 $(110)_2$ 為基底解，翻轉每個位元以產生三個鄰近解。鄰近解中的最佳解 $(010)_2$「不」比 $(110)_2$ 好，演算法結束，近似解為 $(110)_2 = (6)_{10}$。

上例中登高演算法求得的近似解 6 也是該問題的最佳解，因此順利解題。不過如果我們以 $(001)_2$ 作為初始解，就沒有那麼順利了。我們求得的近似解 5 只是個區域最佳解，而非全域最佳解。

$s = (001)_2$ $F((001)_2)=56$ $F((000)_2)=325$ $F((011)_2)=99$ $\boldsymbol{F((101)_2)=600}$	$s = (101)_2$ $\boldsymbol{F((101)_2)=600}$ $F((100)_2)=13$ $F((111)_2)=100$ $F((001)_2)=56$
以 $(001)_2$ 為基底解，翻轉每個位元以產生三個鄰近解。鄰近解中的最佳解 $(101)_2$ 比 $(001)_2$ 好，故取代之。	以 $(101)_2$ 為基底解，翻轉每個位元以產生三個鄰近解。鄰近解中的最佳解 $(111)_2$「不」比 $(101)_2$ 好，演算法結束，近似解為 $(101)_2 = (5)_{10}$。

針對上面的函式，我們可以在登高演算法中嘗試八個不同的初始解，其結果為

初始解	$(000)_2$	$(001)_2$	$(010)_2$	$(011)_2$	$(100)_2$	$(101)_2$	$(110)_2$	$(111)_2$
近似最佳解	6	5	6	6	6	5	6	6

在八次嘗試中，有六次可以順利找到全域最佳解；也就是說，如果我們隨機產生一個初始解，以登高演算法求解上述函式最大化問題的成功率為 6/8 = 75%。

這道試題要求你定義出五個不同的函式 $F(x)$，使得登高演算法展現指定的解題成功率。

(1) 值域的範圍為 $0 \le F(x) \le 10^4$。

(2) 每個函式中，$F(\mathrm{x}) \ne F(y)\ \forall\ x \ne y$。

(3) 每個函式的定義域和要求的搜尋成功率如下：

函式	檔案名稱	定義域 (x 的範圍)	解題成功率	得分
F1	F1.txt	$0 \le x \le 7$	100%	5 分
F2	F2.txt	$0 \le x \le 7$	87.5% (7/8)	5 分
F3	F3.txt	$0 \le x \le 15$	87.5% (14/16)	5 分
F4	F4.txt	$0 \le x \le 31$	100%	5 分
F5	F5.txt	$0 \le x \le 31$	96.875% (31/32)	5 分

(4) 未滿足上述限制者不予計分。

輸出說明

這道試題是一個只需要輸出答案 (Output only) 的題目。請依照題目要求產生五個純文字檔案，檔名如上頁所述。檔案中第 1 行表示 $F(0)$ 的值，第 i 行表示 $F(i-1)$ 的值，以此類推。以題目中的函式為例，

$F(0) = 325$	$F(1) = 56$	$F(2) = 1188$	$F(3) = 99$
$F(4) = 13$	$F(5) = 600$	$F(6) = 9999$	$F(7) = 100$

檔案內容如下：

```
325
56
1188
99
13
600
9999
100
```

試題 3.疊紙盒

問題敘述

小嘉上班的包裝紙盒工廠將舉行年終大摸彩，獎品內容非常豐富；但是由於廠區員工眾多，必須是答對廠長提出的問題之員工才有資格參加摸彩活動。今年廠長出的問題是**「利用有限個長方體紙盒疊出最大正方體」**的益智問題。在此問題中，所有長方體紙盒大小都一樣，但可以任意指定長方體紙盒的長、寬與高(單位是公分)大小；另外，在疊出正方體的過程中，所有長方體紙盒的長、寬、高的方向是一致的，亦即不能隨意對長方體紙盒做出旋轉或翻轉的動作。

聰明的你(妳)，請寫一個程式，幫小嘉計算在廠長給定長方體紙盒的長、寬與高大小，以及總共可供使用的長方體紙盒數目後，所能疊出最大正方體的體積為何？

輸入說明

為四個整數數值，分別是長、寬、高、以及總共的長方體紙盒數目，以空格分隔。其中長、寬、高皆是不會超過10^3公分，總共的長方體紙盒數目不會超過10^9個。

輸出說明

為一整數，代表所能疊出最大正方體的體積(不會超過10^{18}立方公分)，若不能疊出則輸出為 0。註：剩下最少紙盒。

【範例 一】

輸入資料

6 4 3 200

輸出資料

13824

【範例二】

輸入資料

121 99 77 10000000

輸出資料

3543788106936

試題 4. 外星生物進化

問題敘述

有一種很特別的外星原生物，如果 2 個或是 3 個在一起就可以進化合併成為單一的組合生物，進化合併時質量維持守恆，2 個或是 3 個質量相同的組合生物又可以再次進化合併起來，它們自己依循很奇妙的規則進化，進化過程中可以選擇合併也可以選擇不合併。如果一開始有一大群外星原生物，以上述規則快速進化，最後會達到一個穩定狀態，留下一組外星生物(原生物或是組合生物)，很特別的是最後留下的生物中，沒有任何一個生物的質量為其它生物質量的整數倍。我們將這些外星生物標示為[x,y]，表示它經過 x 次二合一以及 y 次三合一進化合併而來，原生物標示為[0,0]。

請寫一個程式，由鍵盤讀入一個小於 2^{31} 的正整數代表一開始外星原生物的總數量，在螢幕上列印出一組進化合併到達穩定狀態時的外星生物？(輸出時請依照 x 由小到大排序，有多組可能性時只需要輸出一組即可，程式最多允許執行時間為 3 秒)

程式執行範例：

輸入	輸出
1	[0,0]
7	[0,1] [2,0]
31	[0,2] [1,1] [4,0] 或是 [0,3] [2,0]
771234	[1,11] [2,10] [3,9] [4,6] [5,5] [8,2] [9,1]
1987654321	[0,19] [1,18] [2,14] [5,12] [6,11] [7,9] [15,2] [18,0]