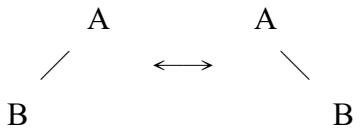
子題 2：後序表示法（post-order）。 **(程式執行限制時間: 2 秒) 15 分**

二元樹的定義：

1.樹不可以為空集合，亦即至少必須有一個根節點，但二元樹卻可以是空集合。

2.樹的兄弟節點位置次序並非固定，但二元樹是固定的。也就是下面是相同的樹，但卻不是相同的二元樹。



在二元樹的運用上，常常需要找出所有的節點資料，這個過程稱為樹的拜訪或追蹤。依拜訪追蹤的次序可分成下列三種：前序表示法（pre-order）、中序表示法（in-order）及後序表示法（post-order）。

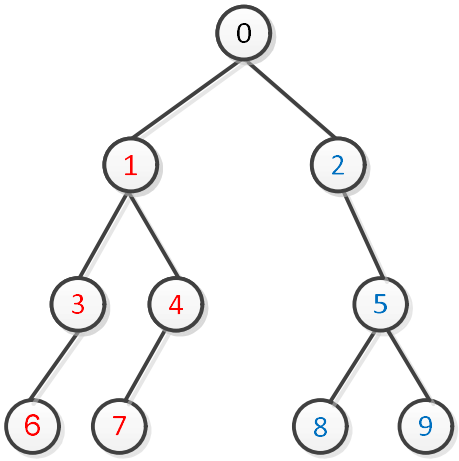
二元樹的走訪（Traversal of Tree）

對於一個二元樹，我們有三種最常用的方法可以走過這棵樹所有的節點。

1. 前序表示法（pre-order）：根節點 -> 左子樹 -> 右子樹
2. 中序表示法（in-order）：左子樹 -> 根節點 -> 右子樹
3. 後序表示法（post-order）：左子樹 -> 右子樹 -> 根節點

有兩種序，就有機會還原出唯一的一棵二元樹。比方說，知道中序表示法（in-order）和前序表示法（pre-order），可以求出原本的二元樹。

在前序表示法（pre-order）之中，最左邊的元素就是 root ；在中序表示法（in-order） 之中， root 的兩邊分別為左子樹和右子樹 —— 利用 root 便可區分左子樹和右子樹。子樹也是樹，可以用相同手法繼續分割，最後便可求出整棵二元樹的架構。



中序表示法（in-order） : 6,3,1,7,4,0,2,8,5,9

前序表示法（pre-order）: 0,1,3,6,4,7,2,5,8,9

輸入說明：

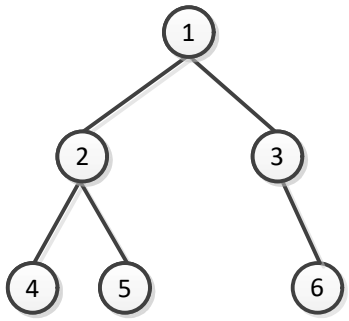
第一列的數字 *n* 代表有幾組資料要測試，$2 \le n \le 8$，第二列起為每組的測試資料，之後每二列為每組的測試資料。每組測試資料的第一列為中序表示法（in-order）；每組測試資料的第二列為前序表示法（pre-order），各節點編號不會相同。測試資料為多個數字x_i，
$0 \le x_i \le 65535$，$4 \le |x_i| \le 64$，中間用逗號隔開。用測試資料找出整棵二元樹的架構。

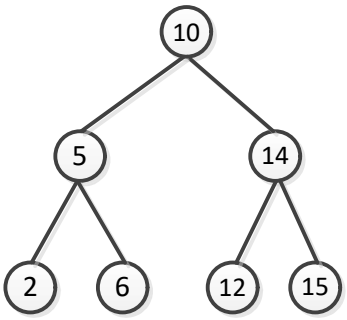
輸出說明：

在測試資料中所建二元樹，輸出這棵二元樹之後序表示法（post-order），每組測試資料輸出一列。

輸入檔案 1：【檔名：in1.txt】

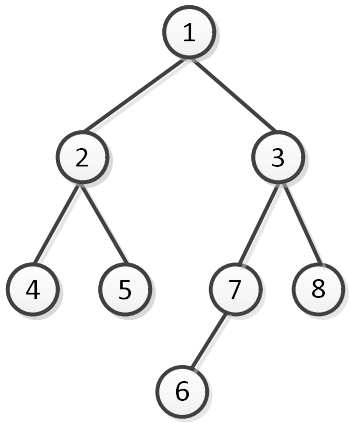
2
4,2,5,1,3,6
1,2,4,5,3,6
2,5,6,10,12,14,15
10,5,2,6,14,12,15

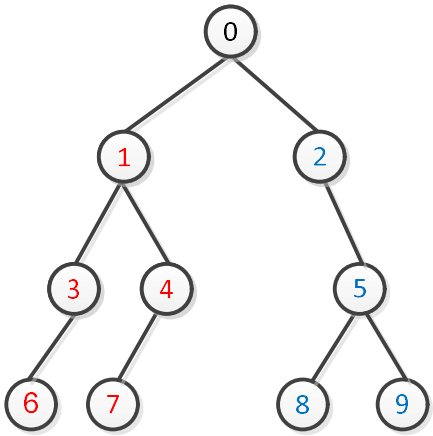




輸入檔案 2：【檔名：in2.txt】

2
4,2,5,1,6,7,3,8
1,2,4,5,3,7,6,8
6,3,1,7,4,0,2,8,5,9
0,1,3,6,4,7,2,5,8,9





輸出範例：【檔名：out1.txt】

4,5,2,6,3,1

2,6,5,12,15,14,10

輸出範例：【檔名：out2.txt】

4,5,2,6,7,8,3,1

6,3,7,4,1,8,9,5,2,0

方便選手比對結果，把上一頁的圖，這也放相同的一份。