Problem 3： 資料結構—樹

子題 1：是否為堆積樹(Heap tree)或二元搜尋樹(Binary search tree)**。(程式執行限制時間: 2 秒)**

14 分

在資料結構中，樹狀結構是可以用來描述有分支的結構，包含 1 個或多個節點。其存在一個特殊的節點，稱為根節點(root)，可連結若干子樹，也可以沒有子樹；從任一節點到根節點，都只有唯一的節點不重複路徑。

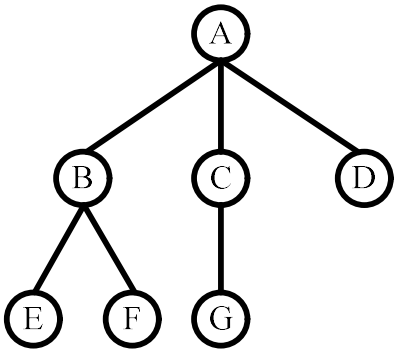


圖 3.1.1

在圖 3.1.1 中，有編號的圓形代表節點，A 為根節點，B、C 及 D 均為 A 的子節點，各節點之間不會有迴圈，且所有節點之間都有一個或多個邊相連通。任一樹狀結構的總邊數等於其總節點數減 1，在樹上任意添加一條邊，就會產生迴圈。

專有名詞介紹：

(1) 無父節點的節點為根節點(Root)，如 A。

(2) 父節點 (Parent)：一節點的上層節點為父節點，如 B 的父節點為 A，如 G 的父節點為 C。

(3) 子節點 (Children)：一節點的下層節點為子節點，如 B 的子節點有 E 及 F；C 的子節點有 G。

(4) 兄弟節點 (Siblings)：有共同父節點的節點稱為兄弟節點，如 B、C、D 互為兄弟節點。

(5) 分支度 (Degree)：一個節點的子樹個數稱為其分支度，如 A 的分支度為 3；B 的分支度為 2；C 的分支度為 1；E 的分支度為 0。

(6) 樹葉節點(Terminal node)：無子節點的節點，如 D、E、F、G。

(7) 內部節點 (Non-terminal node)：樹葉以外的節點均為內部節點，如 A、B、C。

(8) 階層或階度 (Level)：A 為階層 1；B、C、D 為階層 2；E、F、G 為階層 3。

(9) 高度 (Height)：樹的最大階度，例如圖 3.1.1，因最大階度階度為 3，則其樹的高度為 3。

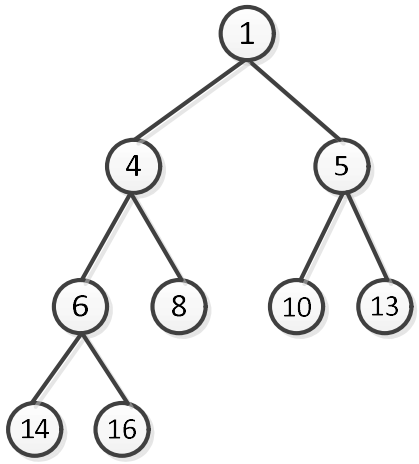
堆積樹(Heap tree)是一個二元樹，每個父節點最多只有兩個子節點，堆積樹的父節點若小於子節點，則稱之為最小堆積（Min heap tree），父節點若大於子節點，則稱之為最大堆積（Max heap tree），而同一層的子節點則無需理會其大小關係。

最小堆積樹(Min heap tree)

指每一個節點的鍵值必須小於它的子節點的鍵值。其特性如下：

1. 每一棵 Min heap tree 是一棵「完整二元樹」(Complete Binary Tree)。
2. 樹根的鍵值小於左子樹與右子樹的鍵值。
3. 其左子樹與右子樹亦是 Min heap tree。如下圖所示：

將下圖的堆積樹轉換為一維陣列之後如下所示：{1,4,5,6,8,10,13,14,16}

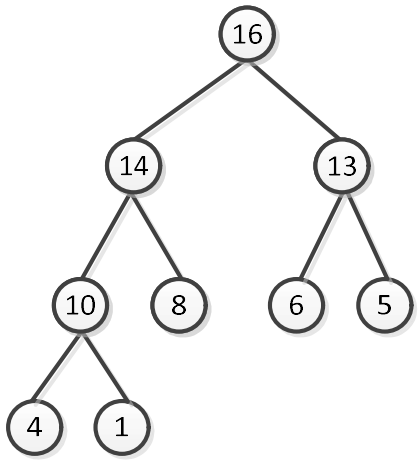


最大堆積樹(Max heap tree)

指每一個節點的鍵值必須大於它的子節點的鍵值。其特性如下：

1. 每一棵 Max heap tree 是一棵「完整二元樹」(Complete Binary Tree)。
2. 樹根的鍵值大於左子樹與右子樹的鍵值。
3. 其左子樹與右子樹亦是 Max heap tree。如下圖所示：

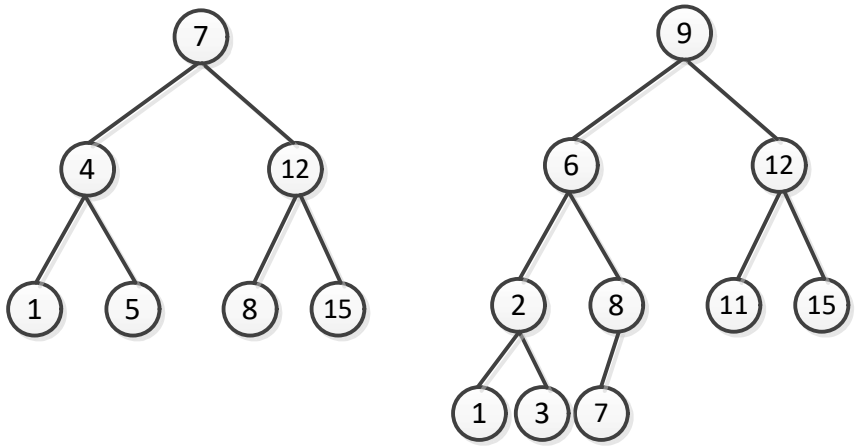
將下圖的堆積樹轉換為一維陣列之後如下所示：{16,14,13,10,8,6,5,4,1}



而在一棵二元樹中，除最後一層外，若其餘層都是滿的，並且最後一層或者是滿的，或者是在右邊缺少連續若干節點，則此二元樹為「完整二元樹」(Complete Binary Tree)。

二元搜尋樹(Binary search tree)定義:
二元搜尋樹是一種二元樹，它可以為空，若不為空，則必須要滿足以下條件：
1.若左子樹不為空，則左子樹的鍵值均須要小於樹根的鍵值。
2.若右子樹不為空，則右子樹的鍵值均須要大於樹根的鍵值。
3.左子樹與右子樹必須也要保持二元搜尋樹。

將下圖的二元搜尋樹(「完整二元樹」(Complete Binary Tree))轉換為一維陣列之後如下所示：{7,4,12,1,5,8,15}和{9,6,12,2,8,11,15,1,3,7}。



寫一個程式，讀入一資料，資料內容為「完整二元樹」(Complete Binary Tree)，然後回答該資料是否為堆積樹(Heap tree)或二元搜尋樹(Binary search tree)。

輸入說明：
第一列的數字 ***n*** 代表共有幾組資料要測試，$2 \le n \le 8$。
第二列起每一列代表一組測試資料。每組測試資料代表一「完整二元樹」(Complete Binary Tree)。測試資料為多個不同的數字x_i，$0 \le x_i \le 65535$，$4 \le |x_i| \le 64$，中間用逗號隔開。

輸出說明：
每組測試資料輸出一列。輸出每組測試資料是否為堆積樹(Heap tree)或二元搜尋樹(Binary search tree)。若該資料是堆積樹(Heap tree)，則輸出 H；若該資料是二元搜尋樹(Binary search tree)，則輸出 B；若不是堆積樹(Heap tree)同時也不是二元搜尋樹(Binary search tree)，則輸出 F。不會有同時為堆積樹(Heap tree)和二元搜尋樹(Binary search tree)的測試資料。

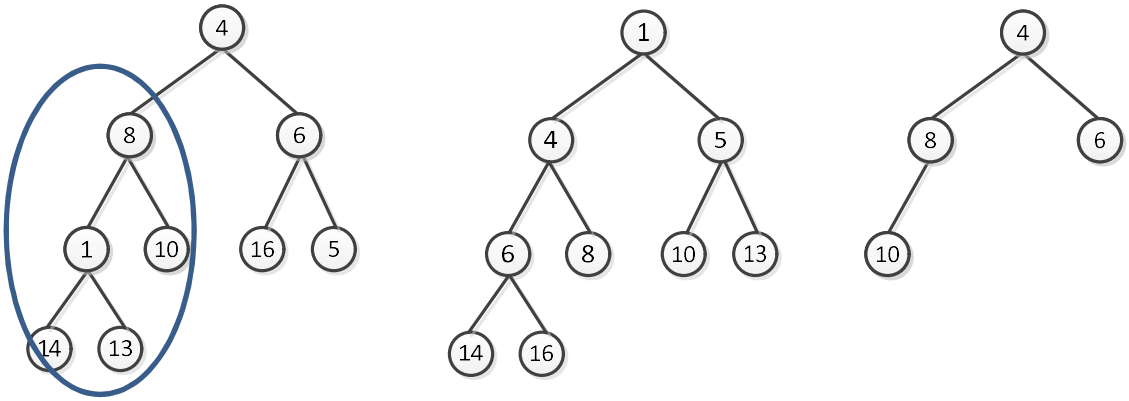
輸入檔案 1：【檔名：in1.txt】

3

4,8,6,1,10,16,5,14,13

1,4,5,6,8,10,13,14,16

4,8,6,10



輸入檔案 2：【檔名：in2.txt】

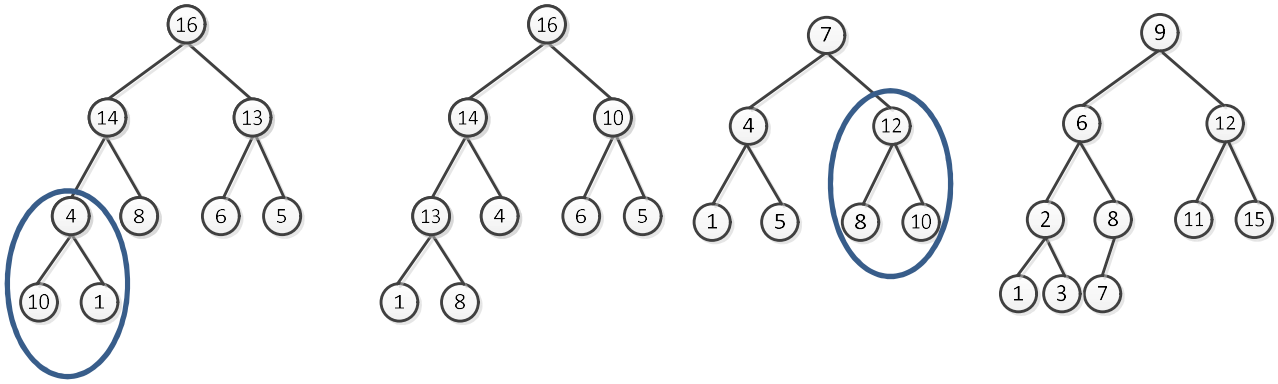
4

16,14,13,4,8,6,5,10,1

16,14,10,13,4,6,5,1,8

7,4,12,1,5,8,10

9,6,12,2,8,11,15,1,3,7



輸出範例：【檔名：out1.txt】

F

H

H

輸出範例：【檔名：out2.txt】

F

H

F

B