《資料結構》

試題評析	本次試題著重在 trees 與 graphs 兩個主要的部分，考生只要準備充足，應可拿到一定的分數。 第一題爲基本的二元樹追蹤問題，拿分相當容易。 第二題則是將迷宮問題改以圖形表示，然後採用 DFS 或 BFS 進行追蹤，小心處理亦可取分。 第三題是二元搜尋樹基本的刪除處理，也應可簡單取分。 第四題是解釋名詞並舉例說明，是很基本的問題。 第五題是測驗樹的結構轉換，需要一些觀察與思考，然後再寫出程式。 綜觀整份試題，最關鍵的應是第五題，因爲須有一些分析與程式設計的能力才能答出。預計一般考生可拿到 70 分左右，準備較充分者可拿到 90 分左右。

一、下列兩節點序列分別為某二元樹之中序追蹤節點序列（in-order）與前序追蹤節點序列（pre-order）。

中序追蹤節點序列：C, B, F, E, G, D, H, A, J, I, L, K, N, M

前序追蹤節點序列：A, B, C, D, E, F, G, H, I, J, K, L, M, N

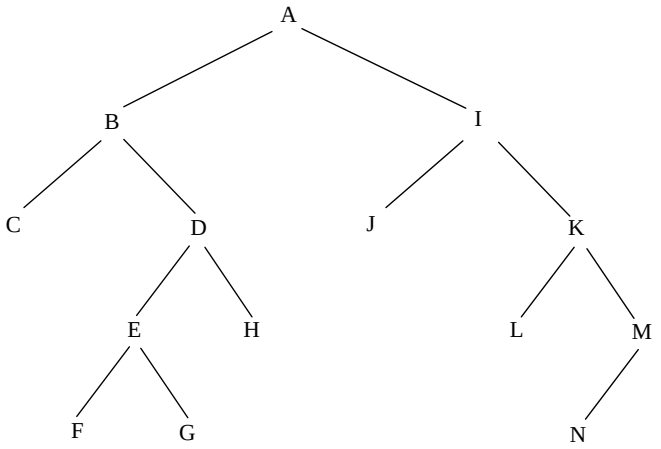
依此兩節點序列可以重建此二元樹。

（一）請逐步演算此重建過程並說明理由。(15 分)

（二）請列出此二元樹的後序追蹤節點序列（post-order）。(5 分)

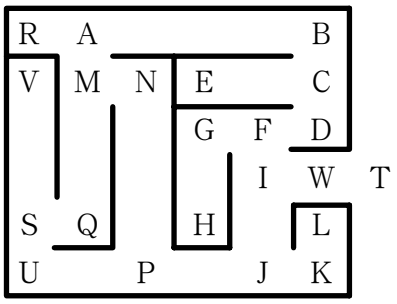
【擬答】

(一) 重建的二元樹如下



(二) 後序追踪節點序列：C,F,G,E,H,D,B,J,L,N,M,K,I,A

二、下圖為一迷宮。將有標記的位置視為一個節點（node），兩節點在迷宮中可以不經由其他節點走到即構成一邊（edge）。



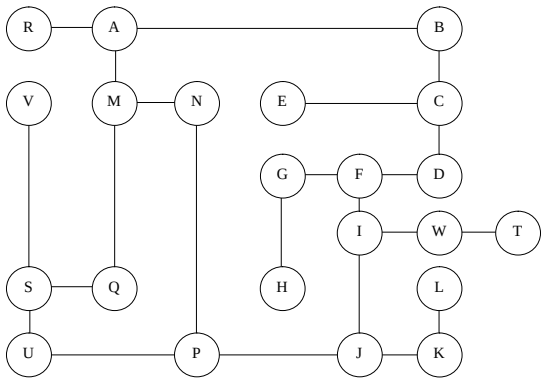
（一）請畫出此圖形（graph）。（2 分）

（二）有一個人在位置 S 上要走出迷宮到達位置 T。此人以先向下、再向左、再向上、再向右的優先順序搜尋節點做深度優先搜尋（depth-first-search）來尋找走出迷宮的路徑。請依時間先後列出其經過的節點。（9 分）

（三）請列出以 S 為出發點以先向下、再向左、再向上、再向右的優先順序搜尋節點的廣度優先搜尋樹（breadth-first-search tree）。（9 分）

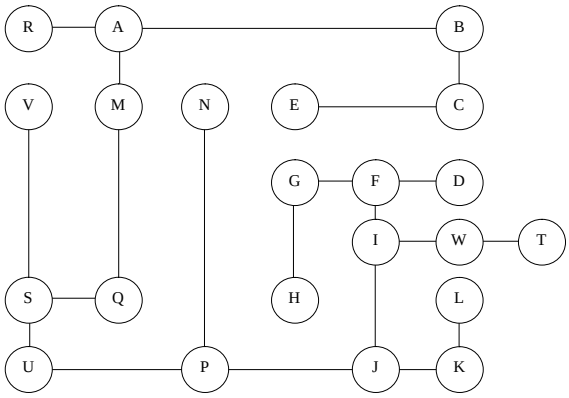
【擬答】

(一)



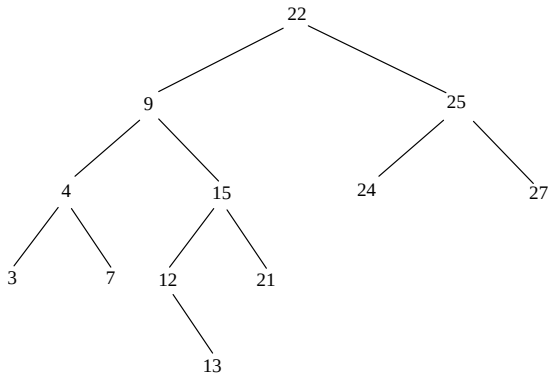
(二) S,U,P,N,M,Q,A,R,B,C,D,F,I,J,K,L,W,T

(三)

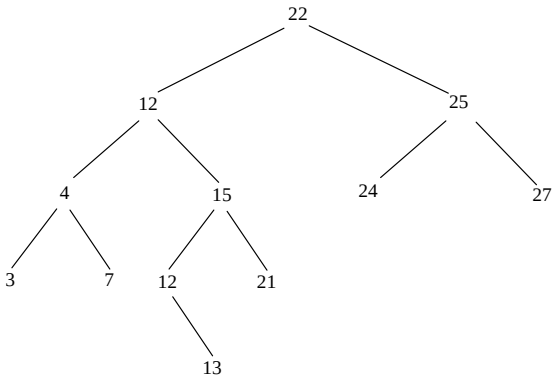


三、將下列十二個鍵值：22, 9, 4, 7, 25, 15, 24, 12, 3, 21, 27, 13
依序插入一空的二元搜尋樹（binary search tree）中。
（一）請列出中間各個步驟的二元搜尋樹。(10 分)
（二）在已插入十二鍵值的二元搜尋樹中，刪去鍵值 9。請列出刪除鍵值 9 的步驟。(10 分)

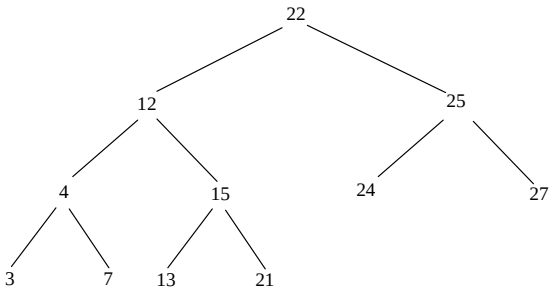
【擬答】
(一)



(二) 將 inorder successor 12 複製到 9 的節點



刪除節點 12，將節點 15 左指標指向節點 13



四、解釋下列名詞，並舉例說明。

（一）最小生成樹（minimum spanning tree）(5 分)

（二）2-3-tree（5 分）

（三）Heap sort（5 分）

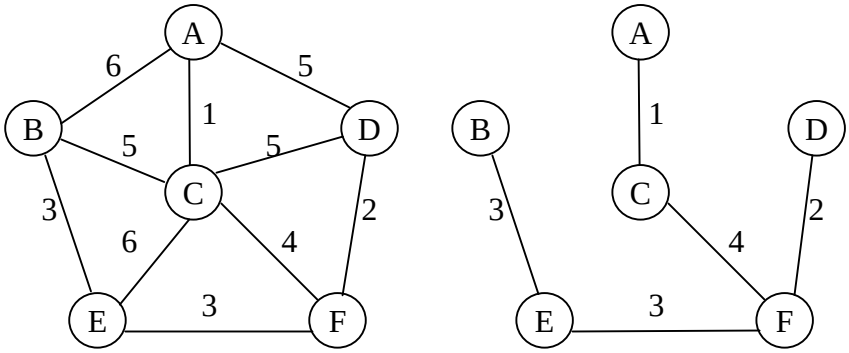
（四）AVL tree（5 分）

【擬答】

(一)最小生成樹

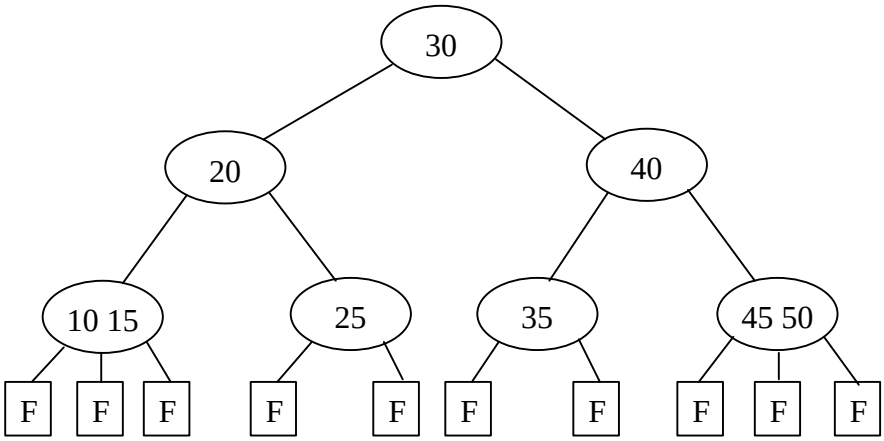
一個圖形的生成樹中，邊的加權值的最小一棵，即爲圖形的最小生成樹。

例如：左圖的 min. spanning tree 爲右圖



(二)2-3 樹

order 爲 3 的 B-tree，每個節點最多有 3 個 subtrees，最少有 2 個 subtrees，而且樹葉皆在且一個 level 的 3-way search tree。例如下圖即爲一棵 2-3 tree：

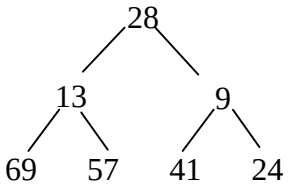


(三)Heap sort

先將全部的資料建立成爲 max heap，再逐步由 max heap 中取出最大的元素，所進行的一種排序。

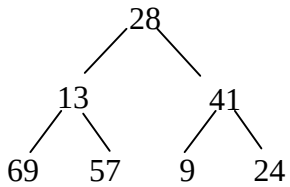
範例：一個 heap sort 的例子

原始資料：28,13,9,69,57,41,24

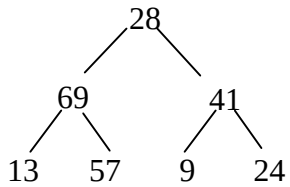


第一階段：建立堆積

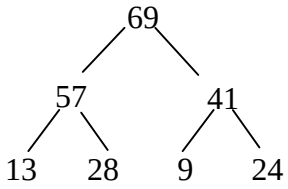
1.由 9 向下進行 sift => 28,13,41,69,57,9,24



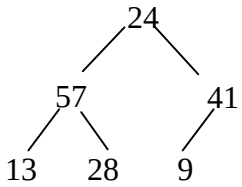
2.由 13 向下進行 sift => 28,69,41,13,57,9,24



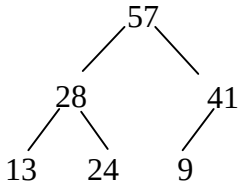
3.由 28 向下進行 sift => 69,57,41,13,28,9,24



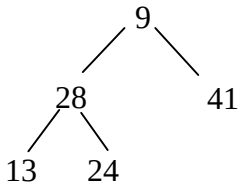
第二階段：堆積排序
1.交換 69 與 24 => 24,57,41,13,28,9,69



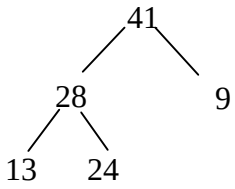
由 24(root) 向下進行 sift => 57,28,41,13,24,9,69



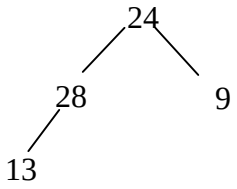
2.交換 57 與 9 => 9,28,41,13,24,57,69



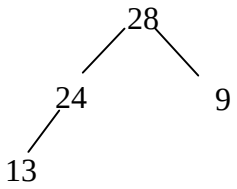
由 9(root) 向下進行 sift =>41,28,9,13,24,57,69



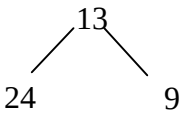
3.交換 41 與 24 => 24,28,9,13,41,57,69



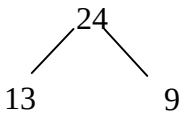
由 24(root) 向下進行 sift =>28,24,9,13,41,57,69



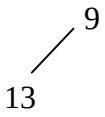
4.交換 28 與 13 => 13,24,9,28,41,57,69



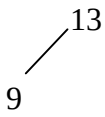
由 13(root) 向下進行 sift =>24,13,9,28,41,57,69



5.交換 24 與 9 =>9,13,24,28,41,57,69



由 9(root) 向下進行 sift =>13,9,24,28,41,57,69



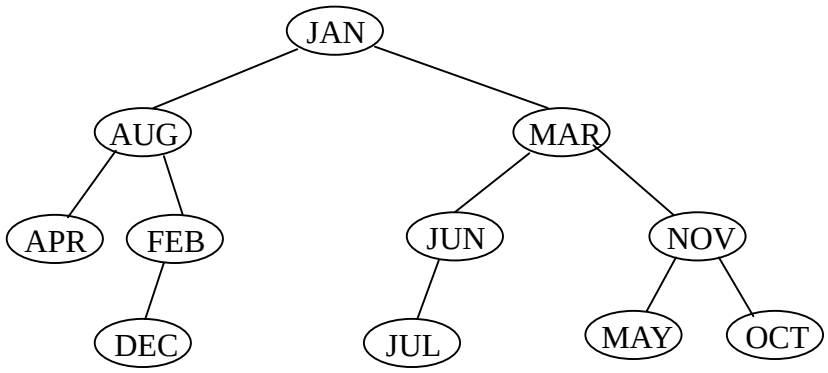
6.交換 13 與 9 =>9,13,24,28,41,57,69 (完成)

9

(四)AVL-Tree

一種高度平衡樹，每個節點的左、右子樹高度，相差不超過 1 的一種二元搜尋樹。

例如：下面爲一棵 AVL tree

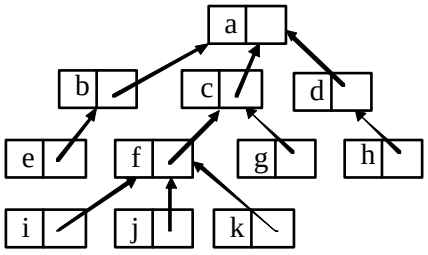


五、以下有兩種資料結構來表示一棵有根樹（rooted tree）：

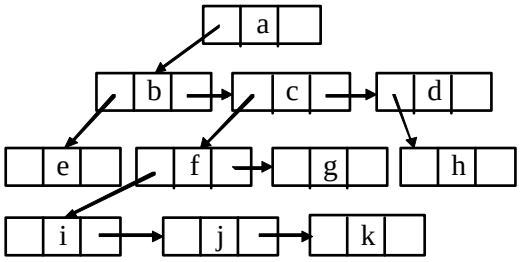
(a)每一個節點（node）有一個指標 par[ ]指向此節點的父節點（parent node）。

(b)每一個節點有兩個指標 first-child[ ]與 sibling[ ]，指標 sibling[ ]指向此節點的一個兄弟節點（sibling node）使得此節點的所有兄弟節點構成一個串列（linked list），而 first-child[ ]指向此節點的所有子節點（child node）由 sibling[ ]所構成的串列的第一個子節點。

請寫一段程式將以資料結構(b)表示的一棵有根樹轉換成以資料結構(a)表示。(20 分)



資料結構(a)範例



資料結構(b)範例

【擬答】

```
Transform (treeptr t)
    {   treeptr child;
        child=first_child[t];
        while (child!=NULL)
        {   par[child]=t;
            Transform(child);
```

```
            child=sibling[child];
        }
}
```

主程式

```
    if (root!=NULL)
      {     Transform(root);
            par[root]=NULL;
      }
```