《資料結構》

試題評析	第一題：測驗合併排序所使用的兩種選擇樹。考前若有準，取得分數不難。 第二題：關節點的相關計算與判定方法，屬於以往較少考的內容，若有複習而取分，相對獲得更多優勢。 第三題：稀疏矩陣的表示法，以及其快速轉置的做法，有詳細準備的考生應可取得分數。 第四題：樹的處理與相關計算，在本試題中，屬於較簡單的題目。 第五題：較少考的內容，而且需要逐步推導，亦屬於較繁複的題目，若考前確實準備而拿到分數，可比其他考生得到相當多的競爭優勢。 綜觀此份試題涵蓋範圍十分廣泛，反應出近年國家考試的命題趨勢，未來考生需做十分周全的準備，方能在資料結構一科，得到應有的分數。
考點命中	第一題：《高點資料結構》，王致強編撰，頁 9-45~9-49。 第二題：《高點資料結構》，王致強編撰，頁 8-52~8-56。 第三題：《高點資料結構》，王致強編撰，頁 3-64~3-66。 第四題：《高點資料結構》，王致強編撰，頁 6-27 精選例題 32、頁 6-33 精選例題 38。 第五題：《高點資料結構》，王致強編撰，頁 11-59~11-65。

一、(一)假設有 8 個排序好的數列(如圖 1)，請建構一 loser tree 並顯示取出前 4 個最小值之 loser tree 變化。(16 分)

(二)請說明 loser tree 和 winner tree 差異為何？(4 分)

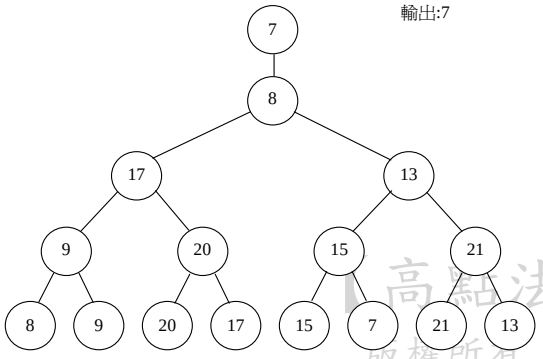
8	9	20	17	15	7	21	13
23	12	70	25	45	10	31	32
27	38	80	28	48	19	36	33

圖 1

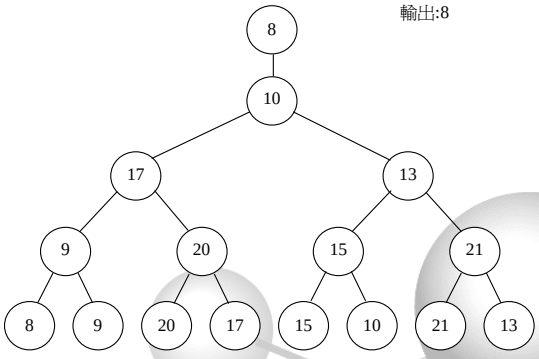
【擬答】

(一)

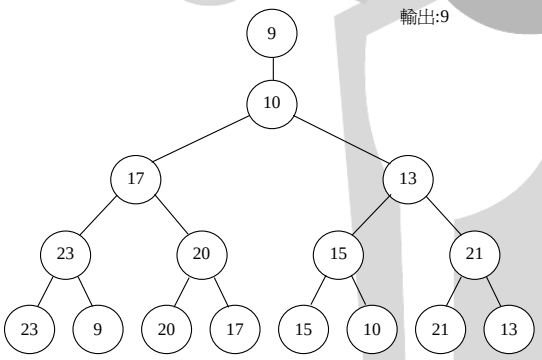
(1)



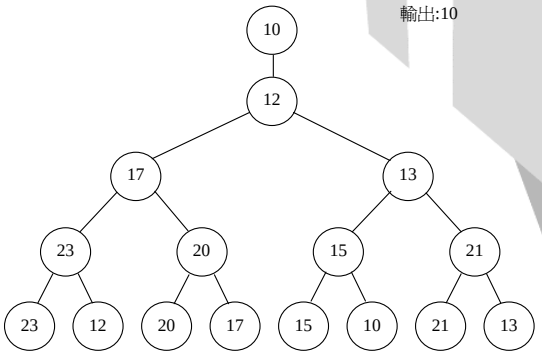
(2)



(3)



(4)



(二) loser’s tree 記錄兩個子節點較大者；而較小者住其父節點移動。
winner’s tree 只記錄兩個子節點較小者；較大者仍停留在子節點。
處理方式如下：
loser’s tree：
每次輸出最小的資料之後，loser’s tree 由最小的資料所在的 run 遞補下一個資料，往 root 的路徑往上一路比較，每經過一個節點就與該節點原有的資料比較，將較大的留在該節點上；而較小的繼續往 root 方向移動，此程序一直到 root 爲止，最後最小的資料在 root 上方的節點。
winner’s tree：
每次輸出最小的資料之後，winner’s tree 也是由最小的資料所在的 run 遞補下一個資料至終端節點，然後與其 sibling 比較，將較小的資料移至父節點，然後上移的資料繼續與 sibling 比較，依此類推，一直到 root 爲止，最後 root 即爲最小的資料。

二、(一)請利用 *dfn*（depth-first number）及 *low*（the lowest depth-first number）值，找出圖 2 所有之關節點（articulation points）。假設利用深度優先搜尋法（depth first search）讀取節點之順序為 4-2-1-3-5-6-8-9-7，也就是節點 4 之 *dfn* 值為 1，節點 2 之 *dfn* 值為 2，節點 1 之 *dfn* 值為 3，依此類推。(15 分)

(二)請說明如何判斷那一節點為關節點？*low* 計算之公式為何？（5 分）

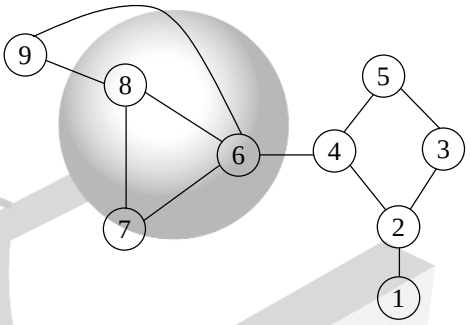
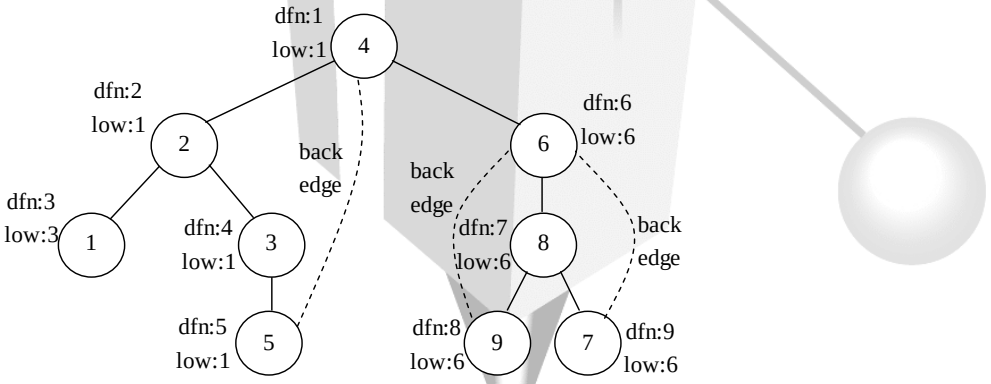


圖 2

【擬答】

(一)dfs spanning tree 如下:



dfn 與 low 值如下表所示：

vertex	1	2	3	4	5	6	7	8	9
dfn	3	2	4	1	5	6	9	7	8
low	3	1	1	1	1	6	6	6	6

articulation points 有 2,4,6 三個。

(二)判斷 articulation point 的規則，是由 dfs spanning tree 中，即據頂點的 dfn 與 low 值來判斷，分爲兩種情況，分別使用不同規則：

1.如果 vertex u 是 root：只要有超過一棵以上的 sub-trees，則 root 即爲 articulation point，如本題的頂點 4。

2.如果 vertex u 不是 root：在 dfs spanning tree 中，如果 u 有任一個子節點 v 的 low[v]≧dfn[u]，則頂點 u 即爲 articulation point，如本題的頂點 2 與頂點 6。

三、給一稀疏矩陣（sparse matrix）M 如圖 3 所示。

(一)請以 3-tuple form（*i, j, value*)來表示此矩陣 M。(6 分)

(二)針對(一)之 3-tuple form，請設計一有效率而時間複雜度不大於 O（*columns* + *terms*）之快速矩陣轉置（fast matrix transposing）演算法。其中 *columns* 為欄的數目，*terms* 為非零項目的數目。以圖 3 所示，*columns* = 4、*terms* = 6。(14 分)

$$\begin{bmatrix} 5 & 0 & 0 & 11 \\ 0 & 41 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 63 & 0 & 23 & 0 \\ 0 & 0 & 0 & 12 \end{bmatrix}$$

圖 3

【擬答】

(一)

	i	j	value
0	5	4	6
1	1	1	5
2	1	4	11
3	2	2	41
4	4	1	63
5	4	3	23
6	5	4	12

(二)使用下面的 fast transpose 方法：

```
1.b[0,i]=a[0,j]; b[0,j]=a[0,i]; b[0,value]=a[0,value];
2.if a[0,value]>0 then begin
3.     for col:=1 to a[0,j] do RowSize[col]:=0;
4.     for t:=1 to a[0,value] do RowSize[a[t,j]]:=RowSize[a[t,j]]+1;
5.     RowStart[1]:=1;
6.     for col:=2 to a[0,j] do RowStart[col]:=RowStart[col-1]+RowSize[col-1];
7.     for t:=1 to a[0,value] do begin
8.           b[RowStart[a[t,j]],i]:=a[t,j];
9.           b[RowStart[a[t,j]],j]:=a[t,i];
10.                b[RowStart[a[t,j]],value]:=a[t,value];
11.                RowStart[a[t,j]]:= RowStart[a[t,j]]+1;
12.          end
13.       end;
```

時間分析：

line 3: O(columns)

line 4: O(terms)

line 6: O(columns)

line 7-12: O(terms)

總時間:O(columns+terms)。

四、(一)請設計一演算法，將一個二元樹（binary tree）每一節點之左子樹和右子樹對調（swap），如下圖 4 所示。(8 分)

(二)假設一 n 個 nodes 之 *k-ary* tree（即分支度為 *k* 之樹）T，每一個 node 有一固定大小之欄位如下，請說明共有多少欄位是 Null？(8 分)

資料	欄位 1	欄位 2	…	欄位 k

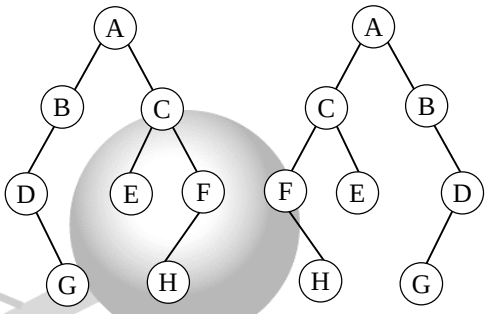


圖 4

【擬答】

(一)

```
1．void swaptree(treeptr t)
2．{     treeptr p;
3．      if (t!=NULL)
4．      {
5．          p=t→left;   t→left=t→right;   t→right=p;
6．          swaptree(t→left);
7．          swaptree(t→right);
8．      }
9．}
```

(二)整棵樹共有 $n \times k$ 個指標欄位，有被用來指向子節點的有 $n-1$ 個指標欄位，因此空指標欄位(即未用到的指標)爲 $n \times k-(n-1)=n \times (k-1)+1$ 個。

五、(一)請說明紅黑樹（red-black tree）之特性。(4 分)

(二)建立一紅黑樹，其數字依序為 10、72、14、68、20、58、30、50、65、63。(10 分)

(三)請一步一步刪除圖 5 紅黑樹之節點，依序為 10、18、3、16、13、12、17。其中在圖 5 之節點 7、12、20 為紅色節點。(10 分)

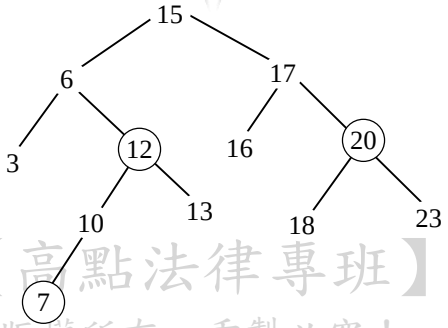


圖 5

【擬答】

(一) red-black tree 是一棵 binary search tree，且必須滿足下面特性：

(1)每個節點可以是 red node 就是 black node。

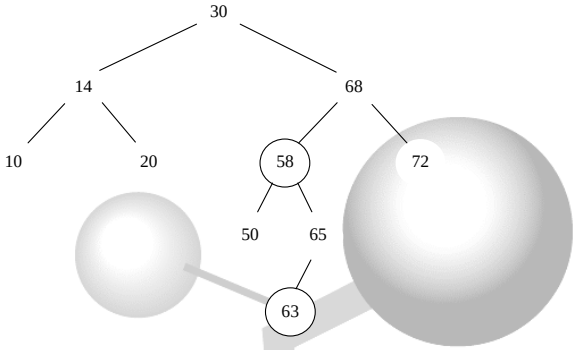
(2)root 一定是 black node。

(3)所有的 leaves(external node)皆爲 black nodes。

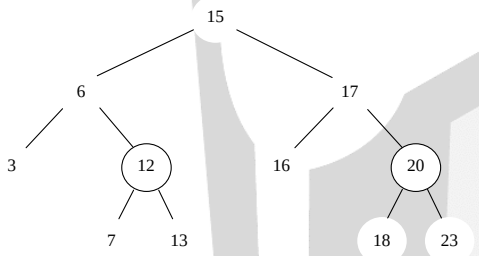
(4)一個 red node，其 children 皆爲 black node。

(5)每個節點到其所有後代 leaves 的每一條路徑上，皆包含相同數目的 black node。

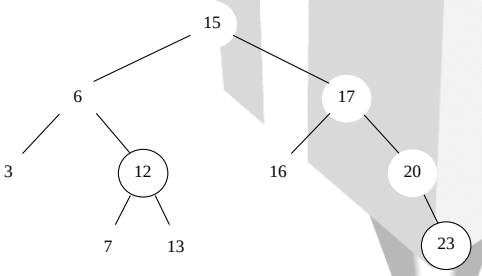
(二)



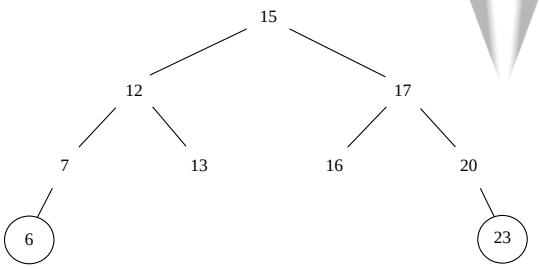
(三)刪除 10



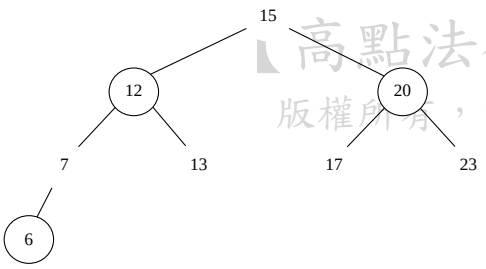
刪除 18



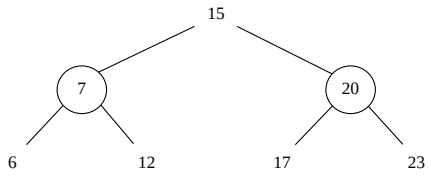
刪除 3



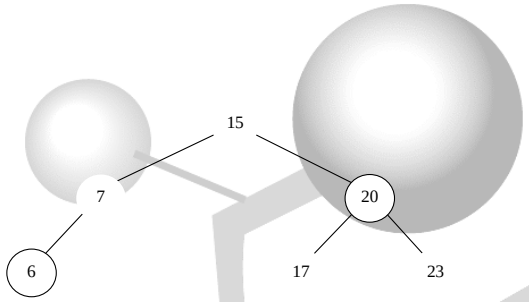
刪除 16



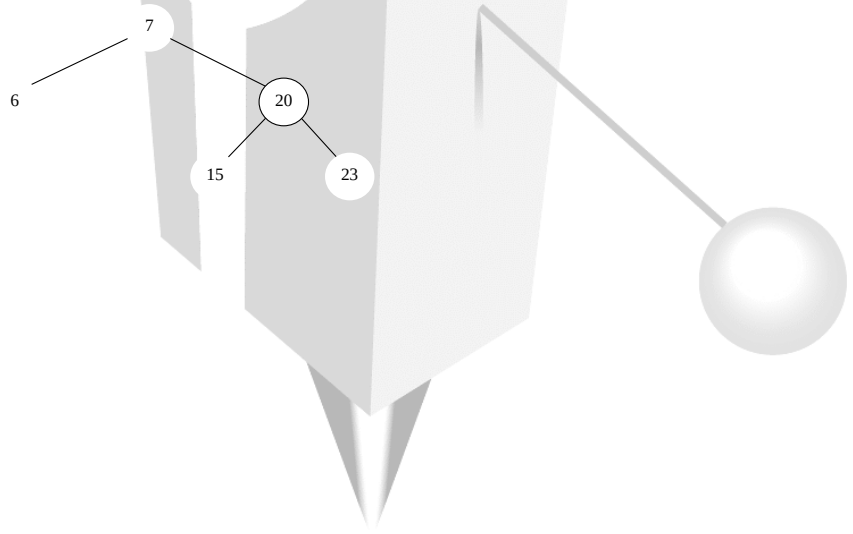
刪除 13



刪除 12



刪除 17



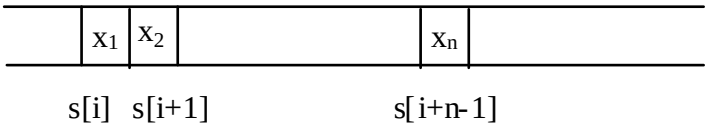
《資料結構》

試題評析	第一題：字串表示法與字串比對問題，爲鏈結串列應用。 第二題：大富翁遊戲的資料結構設計，屬於應用題。 第三題：程式語言物件導向的類別定義，本題應屬於程式語言範圍。 第四題：CPU 排程所需資料結構，屬於應用題。 綜合而言，今年題目並未考到較複雜的資料結構，大多題目偏向應用題型，有些必須具備該應用領域相關知識，才能了解題意，選出適當的資料結構來配合。一般考生的分數可能在 50~60 左右，具備相關應用領域知識者，應可拿到 70~80 分。
考點命中	第一題：《資料結構》，王致強編撰，高點出版，頁 3-72~3-76。 第二題：《資料結構》，王致強編撰，高點出版，第 3 章鏈結串列。 第三題：《資料結構》，王致強編撰，高點出版，頁 3-2~3-3。 第四題：《資料結構》，王致強編撰，高點出版，頁 4-17~7-18、頁 7-3~7-6。

一、「字串比對」是要找出某一有興趣的字串是否包含於另外一個較大的字串或文章中，在字串比對演算法中，我們需要怎麼樣的資料結構來幫助我們求得該字串是否出現？請詳述之。(25 分)

【擬答】

字串比對可以使用 KMP 演算法(Knuth-Morris-Pratt)效率較佳，配合使用的資料結構爲連續記憶體(如一維陣列)較佳，以方便配合 KML 演算法。



(1) 先對 pattern 找出其 prefix function(或稱爲 failure function) f。

f(j)的求法：

f(j) = 最大的 i<j，使得 $p_1p_2\cdots p_i=p_{j-i+1}p_{j-i+2}\cdots p_j$, 存在有 i≧1；

f(j) = 0, 其他情況。

(2) KMP 演算法進行字串比對

```
q=0;
for (i=1;i<=n;i++)
{     while (q>0 && p[q+1]!=s[i]) q=f[q];
      if (p[q+1]==s[i]) q++;
      if (q==m) {    printf("pattern occurs at %d\n", i-m+1);
                     q=f[q];
                }
}
```

時間複雜度爲 O(m+n)，其中較長的字串長度爲 n；較短字串長度爲 m。

二、在大富翁的遊戲中，我們需要那些資料結構來幫助我們設計該遊戲？請詳述之。(25 分)

說明：大富翁（Monopoly）是一種多人策略圖版遊戲。參賽者分得遊戲金錢，憑運氣（擲骰子）及交易策略，買地、建樓以賺取租金。

【擬答】

(一)Monopoly 有一個盤面，通常是一個方形環狀的道路，此一盤面使用環狀的鏈結串列來實作，讓代表玩家的棋子沿著鏈結串列循環前進即可。

(二)每個位置有路名、所屬玩家名稱、是否有建房屋等等資訊，記錄在鏈結串列的節點中，每個串列節點有資料欄位記錄前述資料。

(三)機會與命運兩疊卡片，分別各用一個鏈結串列來表示，每次翻卡片時，就由串列開頭取得第一個節點。

(四)骰子則以亂數函數實作即可。

三、 Point *p* 是 XY 平面中的某一點，由 x 及 y 的座標所組成如下：(x, y)，請用物件的方式寫出 Class Point 並利用 Class Point 進一步定義出 Class Line，Line 是由 XY 平面中的兩點所組成的線段。(25 分)

【擬答】

```
class Point {
   private:
        float x, y;
   public:
        Point(int x, int y)
        {     this.x=x;    this.y=y;    }
}

class Line {
     private:
          Point *point1, *point2;
     public:
          Line(point *p1, point *p2)
          {     point1=new Point(p1->x,p1->y);
                point2=new Point(p2->x,p2->y);
          }
}
```

四、 在作業系統資源管理程式中，我們需要那些資料結構來幫助我們適當的分配 CPU 的執行時間？請詳述之。(25 分)

【擬答】

CPU 排程(scheduling)需視系統採用何種排程演算法，才能決定使用那一種資料結構，以下列舉各種排程法說明：

(一)若使用先到先服務(FCFS, First-Come-First-Serve)，可以使用佇列(queue)的 FIFO 資料結構，即可以配合。

(二)若使用最短工作優先(SJF, Shortest Job First)，可以使用優先權佇列(priority queue)資料結構，實作上是以最小堆積(Min-Heap)，提供新到達工作加入隊伍的插入(Insertion) 與選取最短工作的 Delete-Min 兩個運算，時間複雜度皆爲 O(logn)。

(三)若使用優先權排程(Priority Scheduling)，也是使用優先權佇列(priority queue)資料結構，實作與 SJF 類似。

(四)若使用循迴式排程(Round-Robin Scheduling)，佇列(queue)的 FIFO 資料結構，即可以配合。

《資料結構》

試題評析	第一題：堆疊用於處理運算式，包括中序式轉換後序式，以及後序式求值計算，屬於基本問題。 第二題：各種排序法操作問題，基本上拿分也不難。 第三題：二元樹基本特性與中序追蹤，亦屬基本題型。 第四題：Kruskal algorithm 以及實作所需之資料結構，對演算法有較深入認識，可以得到分數。 綜合來說今年題目平易近人，涵蓋度亦適中，一般應試者可以拿 75~80 分，詳細準備者可拿 90 分以上的分數。
高分閱讀	第一題：王致強，《資料結構》，高點出版，pp.4-55~63。 第二題：王致強，《資料結構》，高點出版，pp.9-63~66、pp.9-51~55、pp.9-24~27。 第三題：王致強，《資料結構》，高點出版，pp.6-12~13(精選範例 9)、p.6-16。 第四題：王致強，《資料結構》，高點出版，pp.8-39~41。

一、(一)請將中序運算式(8×3-6/2)+5/(1+4)轉換成後序運算式（postfix expression）。(10 分)
(二)請使用堆疊（stack）說明算出後序運算式 1, 2, 3, *, 4, 6, +, 5, /, /, +的過程與結果。(10 分)

【擬答】

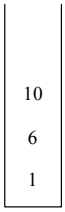
(一)後序運算式爲 8 3 × 6 2 / - 5 1 4 + / +

(二)1,2,3 :存入堆疊。

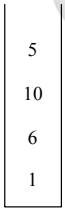
*： 取出兩個運算元做乘法 2 * 3 = 6 存入堆疊。

4, 6 :存入堆疊。

+ : 取出兩個運算元做加法 4 + 6 = 10 存入堆疊。

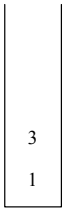


5 :存入堆疊。

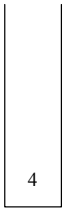


/ : 取出兩個運算元做除法 10 / 5 = 2 存入堆疊。

/ : 取出兩個運算元做除法 6 / 2 = 3 存入堆疊。



+ : 取出兩個運算元做加法 1 + 3 = 4 存入堆疊。



最後，由堆疊取出計算結果 4。

二、有一陣列 A=(163, 231, 356, 93, 869, 987, 58, 349, 271, 33)要由小排到大。

(一)使用基數排序法（radix sort）需要三個回合（pass）排序 A 陣列，請寫出前兩個回合結束時 A 陣列的內容。(10 分)

(二)使用堆積排序法(heap sort)需要先將 A 陣列整理成 maxheap，然後再經過九個回合(pass)的 reheap 才能將資料由小排到大，請寫出整理成 maxheap 後與第一個回合 reheap 結束時 A 陣列的內容。(10 分)

(三)使用快速排序法（quick sort）將 A 陣列排序，每一回合（pass）選擇待排序子陣列（sub-array）最左邊那筆資料做為比較基準，且左邊子陣列會比右半子陣列先處理，請寫出前兩個回合結束時 A 陣列的內容。(10 分)

【擬答】

(一)基數排序法（radix sort）

第一回合

A=(231,271,163,93,33,356,987,58,869,349)

第二回合

A=(231,33,349,356,58,163,869,271,987,93)

(二)堆積排序法（heap sort）

整理成 maxheap 後

A=(987,869,356,349,231,163,58,93,271,33)

第一個回合 reheap 結束時

A=(869,349,356,271,231,163,58,93,33,987)

(三)快速排序法（quick sort）

第一個回合，以 163 爲基準

A=(33,58,93,163,869,987,356,349,271,231)

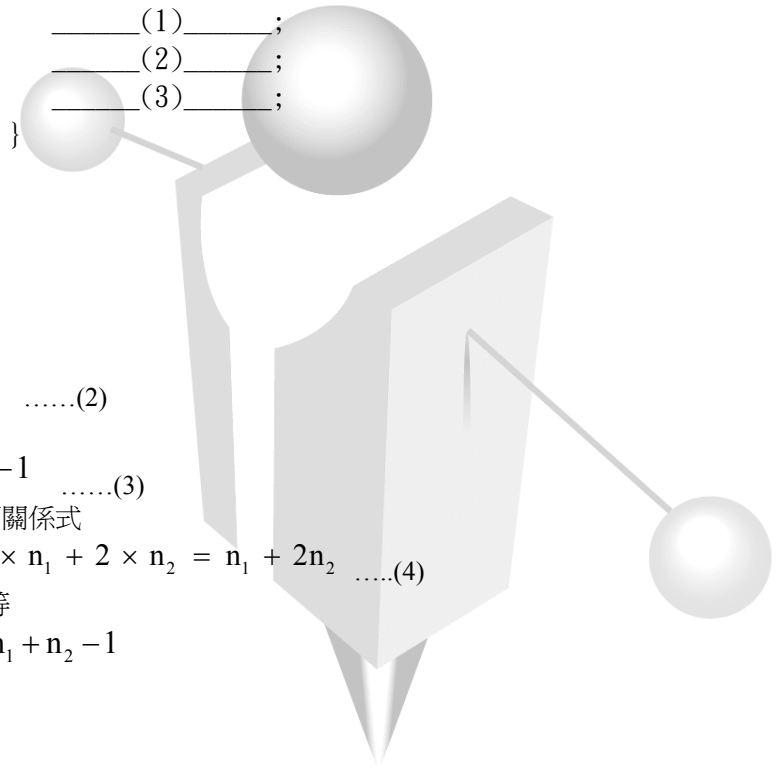
第二回合，以 33 爲基準

A=(33,58,93,163,869,987,356,349,271,231)

三、(一)有一 N 個節點（node）的二元樹（binary tree），令 N_0 代表沒有子節點的樹葉（leaf node）個數，N_1 代表只有一個子節點的節點個數，N_2 代表有兩個子節點的節點個數，請證明 $N_0=N_2+1$。(10 分)

(二)請填入下面 C 程式中三個空格以完成 ptr 指向樹根的二元樹中序追蹤(inorder traversal)程式並將追蹤結果顯示在螢幕上。(15 分)

```
struct node {
struct node  *left;
int          data;
struct node  *right;};
void inorder(struct node *ptr)
{        if(ptr !=NULL){
            ______(1)______;
            ______(2)______;
            ______(3)______;
         }
}
```



【擬答】

(一)證明如下

$b = n - 1$ ……(1)

$n = n_0 + n_1 + n_2$ ……(2)

(2) 代入 (1) 得

$b = n_0 + n_1 + n_2 - 1$ ……(3)

又分支度亦有下面關係式

$b = 0 \times n_0 + 1 \times n_1 + 2 \times n_2 = n_1 + 2n_2$ …..(4)

(3) 與 (4) 應相等

$n_1 + 2n_2 = n_0 + n_1 + n_2 - 1$

化簡即得

$n_0 = n_2 + 1$ 。

(二)

(1) inorder(ptr→left)
(2) printf(“%d”,ptr→data)
(3) inorder(ptr→right)

四、(一)請寫出在無向圖中找出 Minimum Cost Spanning Tree 的 Kruskal 演算法。(15 分)
(二)請說明 heap(除了 heap sort 外)與 disjoint set 這兩種資料結構在這個演算法中有何作用?(10 分)

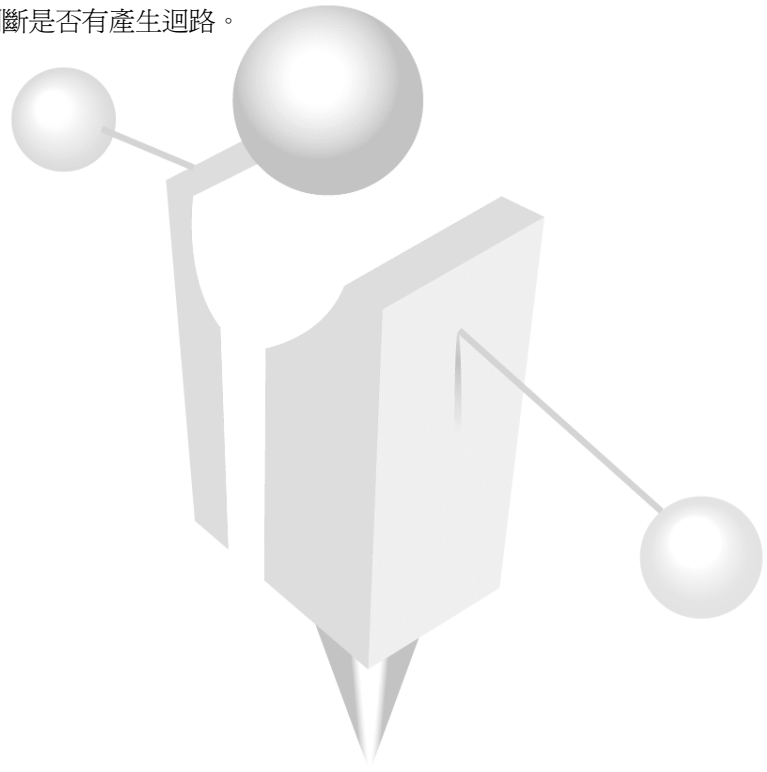
【擬答】

(一)Kruskal 演算法

T←Φ;
while T 中少於 n-1 個 edges 且 E is not empty do
begin

```
(v,w)←E 中 weight 最低的 edge;
delete (v,w) from E;
if (v,w) 不會造成 T 中的迴路 then add (v,w) to T
                             else discard (v,w)
end;
if T 中的 edges 數少於 n-1 then not_found;
```

(二)heap 用來選取 E 中 weight 最小的邊。
disjoint sets 用來判斷是否有產生迴路。



《資料結構》

試題評析	今年考題重點著重在基本觀念與應用。 第一題：雜湊表的觀念，對考生而言並不難，但能否完整詳述雜湊表的各項要點，變成是取分的關鍵。 第二題：本題爲應用題，需要使用到遊戲樹的觀念，如果設計過人工智慧遊戲，則本題不致於太難。 第三題：互斥集合是資料結構中，一種著名的結構，其應用於最低成本伸展樹，亦爲重要的應用範例，因此考生本題應可取得不錯的分數。 第四題：使用陣列來模擬指標或鏈結串列的效果，爲基本的做法，取分亦不難。 綜觀今年考題，大多偏向申論式的問題，未有演算法或程式性的問題，考驗考生的組織能力，是否能完整詳述整體的做法，成爲分數高低的關鍵。預估一般考生應可取得 60~70 分，準備完整且組織能力佳者，應可得到 80 分或更高的分數。
	1.王致強「資料結構」上課講義第六回 PP. 10~13。 2.王致強「資料結構」上課講義第三回 PP. 50~52。 3.王致強「資料結構」上課講義第三回 PP. 44~45、第四回 P. 21。 4.王致強「資料結構」上課講義第二回 P. 61。

一、(一)請解釋 Hash function。其主要功能及設計考量點為何？（15 分）
(二)我們可以用那一種資料結構來實現它？（10 分）

【擬答】

(一)

1.Hash function 是採用一個計算位址的函數，將關鍵值以參數代入，用來計算出資料所應儲存的位置或索引。

2.其功能爲可以在 O(1)(常數時間)存取資料，包括插入新資料，搜尋資料等。

3.設計要點：

(1)碰撞機率：選擇適當的 Hash function 以減少碰撞機率。

(2)表格空間利用率：配置較大空間可以降低碰撞機率，但浪費空間；較小的的表格空間，較不會浪費空間，但碰撞機率相對會較高。

(3)滿溢處理(overflow handling)：當插入新資料時，若 bucket 已滿，如何找到空間存放新資料，分爲開放位址(open addressing)與獨立串列(separate chaining)兩類。

(4)減少群聚(clustering)：滿溢處理要注意儘量不要有 primary clustering 與 secondary clustering 現象發生。

(二)Hash function 使用陣列來實作，一個陣列分以分成 m 個 buckets，每個 bucket 可以包含數個 slots，每個 slot 可以儲存一筆資料。以 hash function 計算出 bucket 編號，然後存取 bucket，插入資料時，若 bucket 已滿，則使用滿溢處理以儲存資料：

1.開放定址法：就在表格中，以特定規則找到別一個未滿的 bucket 儲存資料。

2.獨立串列：則以鏈結串列將新資料建立一個節點存放。

二、在圍棋程式中，最常用的三種資料結構為何？請說明其用途。（30 分）

【擬答】

(一)圖形：記錄已落子的棋子位置，通常用鄰接矩陣(2-D 陣列)記錄棋子位置即可，此一陣列用記錄盤面狀況，來檢查死活，計算佔領區域的大小，以判斷勝負。

(二)遊戲樹：在人工智慧部份用來找尋電腦的最佳落子位置，通常使用 mini-max 演算法或 Alpha-beta pruning 演算法，來找尋最佳落子位置。

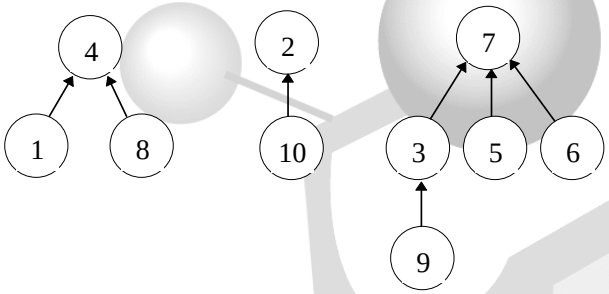
(三)堆疊或佇列：用 backtracking 搜尋遊戲樹時，需要用到堆疊；用廣度搜尋法時，使用佇列；若使用最佳優先搜尋法(best-first search)時，使用優先權佇列(priority queue)。

三、(一)請解釋 disjoint-set data structure。(10 分)
(二)請舉出一個應用的例子。我們可以用那一種資料結構來實現它？(15 分)

【擬答】

(一)disjoint-set 是一群集合，兩兩之間的交集皆爲空集合，則這群集合稱爲互斥集合。通常需要兩個運算。
1.聯集(union)：將兩個互斥集合合併爲一個集合。
2.找尋(find)：找尋元素所屬的集合。

(二)應用範例：在 Kruskal 演算法中，找尋最低成本伸展樹，可以用 disjoint- set 來協助檢查是有迴圈產生。disjoint-set 可以使用 trees 來實現，例. A={1,4,8} B={2,10} C={3,5,6,7,9}爲三個互斥集合，每一個互斥集合用一棵樹來代表。



資料結構
(1)parent[i]≧0：代表指向父親的指標。
(2)parent[i]＜0：表示 node i 爲 root，且整棵樹共有 –parent[i] 個 nodes。

運算：
(1)集合的聯集(Union)：合併兩棵樹,O(1)

```
1.union(x, y)
2.{
3.if (parent[x]<=parent[y])
4.{
5. parent[x]←parent[x]+parent[y];
6. parent[y]←x;
7. }
8. else
9. {
10. parent[y]←parent[y]+parent[x];
11. parent[x]←y;
12.}
13.}
```

(2)元素所屬於集合之找尋(Find)：O(logn)

```
1.find(x)
2.{
3. i←x;
4. while (parent[i]>=0)
5. i←parent[i];
6.// Collapsing
7. j←x;
8.while (parent[j]>=0)
9.{
10. k←parent[j];
11. parent[j]←i;
12. j←k;
```

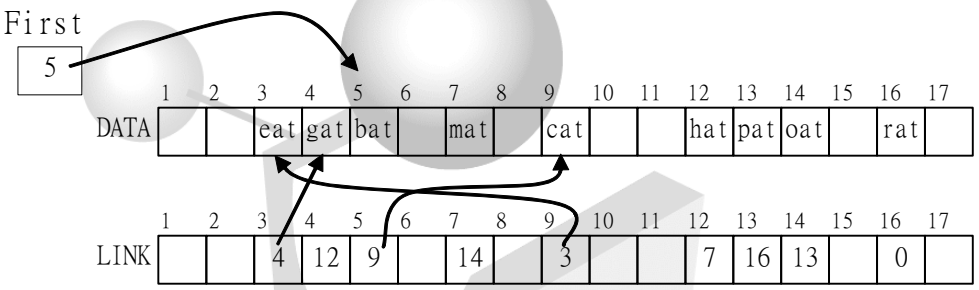
```
13. }
14. return i;
15.}
```

四、我們如何在一個沒有支援 pointer 的程式語言中，利用那一種資料結構來實現 pointer？請舉例說明。(20 分)

【擬答】

以陣列來實作：使用兩個陣列，一個存放資料，另一個指出下一項資料的位置(註標)。

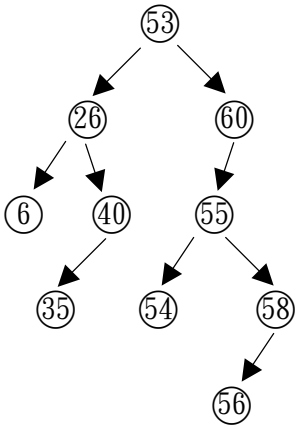
```
Datatype DATA[MaxItem];
int      LINK[MaxItem];
```



《資料結構》

試題評析	1.今年試題十分平易近人，考生只要充分準備，可以取得不錯的分數。考題範圍主要集中在二元搜尋樹、堆積、遞迴、圖形表示法與雜湊法，並未出現特別的難題。 2.第一題考搜尋樹的基本操作，如插入、刪除以及二元樹的追蹤。第二題則是最大堆積的插入與刪除兩個運算。第三題是簡單的遞迴結果的追蹤與時間分析。第四題考圖形的表示法與路徑找尋。第五題是雜湊法的插入與搜尋等問題。 3.綜觀此份試題無艱澀之處，一般考生可以拿到 80 分以上，程度好的同學要拿到 90 分以上亦非難事。

一、圖一為一個二元搜尋樹（binary search tree），每個節點含有一個整數。



圖一

(一)請將 48 加入圖一，並將結果的二元搜尋樹畫出。(5 分)

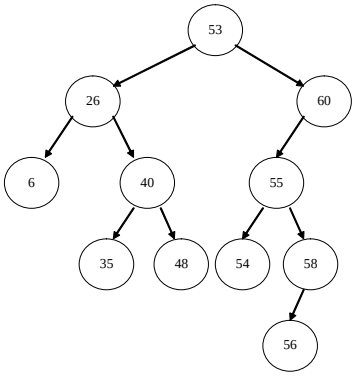
(二)請將 53 從圖一刪除。假設每個數刪除後，皆由小於但最接近的數取代。請將結果的二元搜尋樹畫出。(5 分)

(三)請將圖一以前序（preorder）方式表示。(5 分)

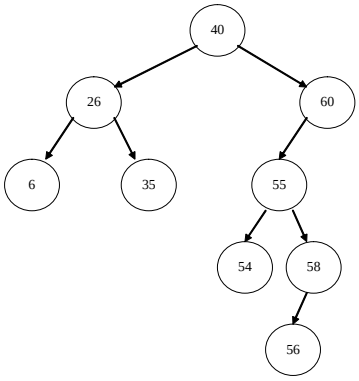
(四)請將圖一以後序（postorder）方式表示。(5 分)

【擬答】

(一)



(二)



(三) 53,26,6,40,35,60,55,54,58,56
(四) 6,35,40,26,54,56,58,55,60,53

二、有兩個陣列 A＝(40, 32, 27, 16, 29, 23, 5, 14, 2, 3, 15)，
B＝(39, 32, 28, 16, 15, 2, 14, 3, 5, 17, 24)。
(一)A 和 B 那一個是最大堆積（maxheap）？（10 分）
(二)將 30 加入是最大堆積的那個陣列中，並將結果的最大堆積以陣列的方式列出。(5 分)
(三)將最大的數從最大堆積的那個陣列中刪除，並將結果的最大堆積以陣列的方式列出。(5 分)

【擬答】
(一)A 是最大堆積
(二)40,32,30,16,29,27,5,14,2,3,15,23
(三)32,29,27,16,15,23,5,14,2,3

三、假設有一個 C 程式：

```
int ppp (int d){
    if (d<=1)
        return d;
    return 2*ppp(d-1)+3*ppp(d-2);
```

}

(一)請問呼叫 ppp(4)的回傳值為何？（10 分）

(二)請問在執行 ppp(4)的過程中，ppp(0)被呼叫幾次？ppp(1)被呼叫幾次？（10 分）

【擬答】

(一)ppp(4)=20

(二)ppp(0)被呼叫 2 次；ppp(1)被呼叫 3 次。

四、有一個鄰接矩陣（adjacency matrix）：

	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	1	0	0	1
C	0	0	0	1	1	0
D	0	0	0	0	0	0
E	0	0	0	1	0	1
F	0	0	0	0	0	0

其中 A,B,C,D,E,F 代表節點（node）。

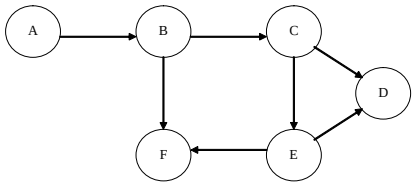
(一)請將對應此矩陣的有向圖型（directed graph）畫出。(5 分)

(二)請將此圖型的鄰接表單（adjacency list）畫出。節點請依字母順序由小到大列出。(5 分)

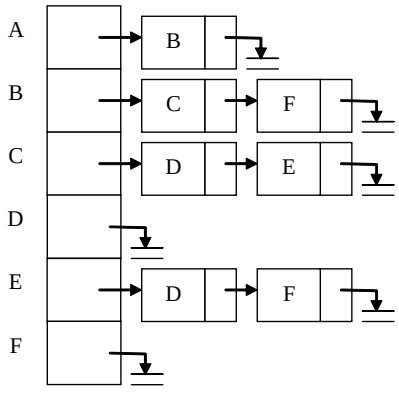
(三)請列出長度為 2 的所有路徑。(10 分)

【擬答】

(一)



(二)



(三)

A,B,C
A,B,F
B,C,D
B,C,E
C,E,F
共五條

五、有一個雜湊表（hash table），共有 11 個籃子（bucket），且每個籃子中可存一個鍵值（key），假設雜湊函數為 h(x)=x%11，亦即除以 11 的餘數。今有 8 個鍵值；73, 25, 29, 33, 51, 41, 20, 43。

(一)請將此 8 個鍵值依次存入此雜湊表，並將結果的雜湊表畫出。假設利用線性探測法（linear probing）來處理碰撞（collision）的問題。(10 分)

(二)假設現在要找鍵值 43，請問需要做幾次鍵值的比較才能找到 43？(5 分)

(三)假設現在要找鍵值 64，請問需要做幾次鍵值的比較才能確定 64 不在雜湊表裡？(5 分)

【擬答】

(一)

0	1	2	3	4	5	6	7	8	9	10
33	20	43	25				73	29	51	41

(二)4 次

(三)6 次(檢查 7 個 buckets)

《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

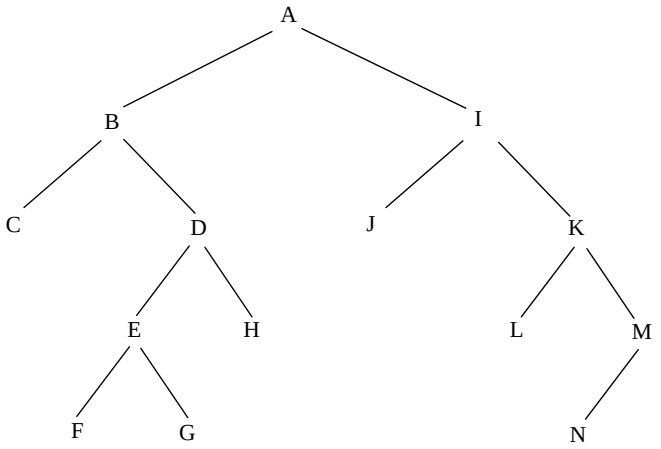
依此兩節點序列可以重建此二元樹。

（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)

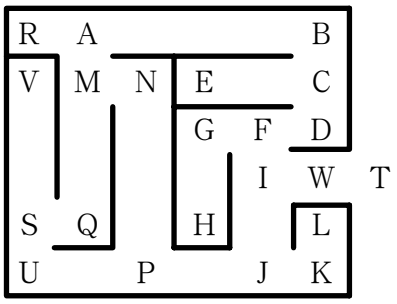
【擬答】

(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）。



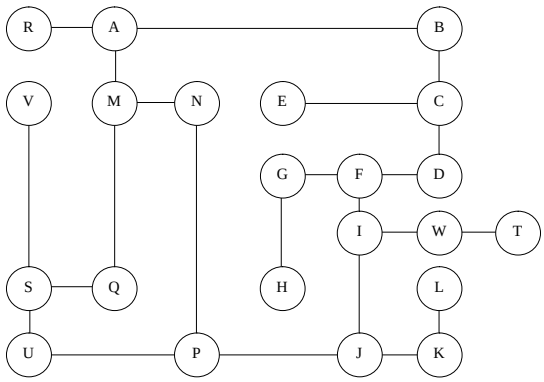
（一）請畫出此圖形（graph）。（2 分）

（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）

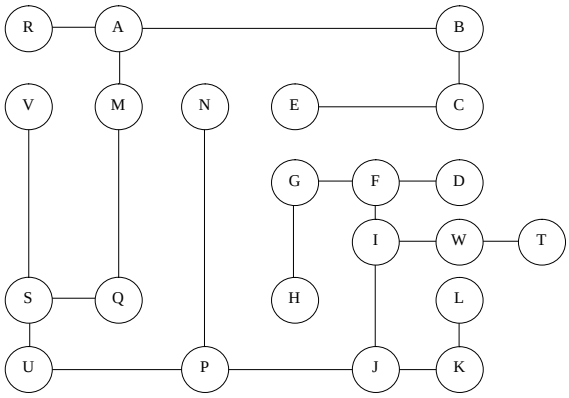
【擬答】

(一)



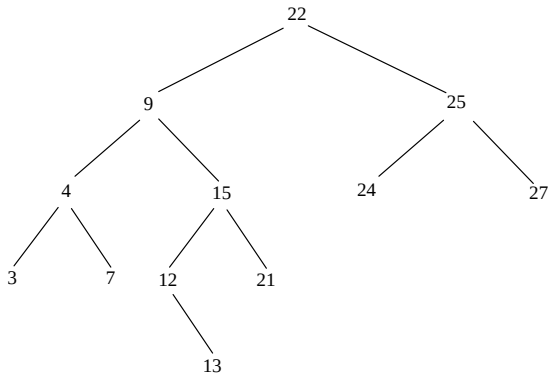
(二) S,U,P,N,M,Q,A,R,B,C,D,F,I,J,K,L,W,T

(三)

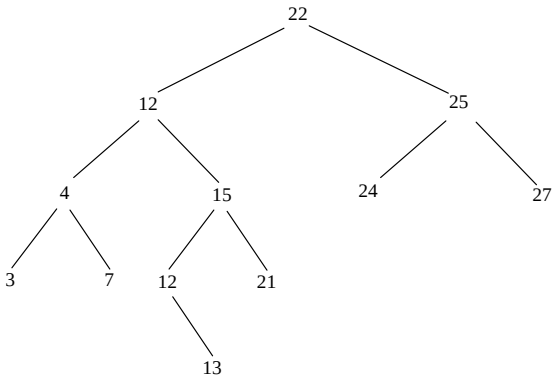


三、將下列十二個鍵值：22, 9, 4, 7, 25, 15, 24, 12, 3, 21, 27, 13
依序插入一空的二元搜尋樹（binary search tree）中。
（一）請列出中間各個步驟的二元搜尋樹。(10 分)
（二）在已插入十二鍵值的二元搜尋樹中，刪去鍵值 9。請列出刪除鍵值 9 的步驟。(10 分)

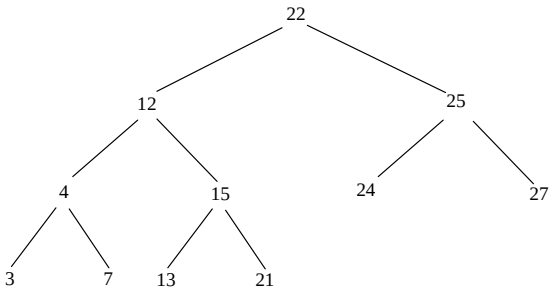
【擬答】
(一)



(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）

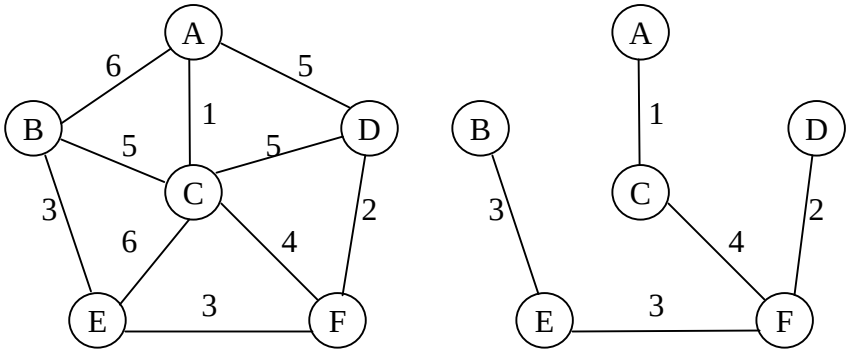
（四）AVL tree（5 分）

【擬答】

(一)最小生成樹

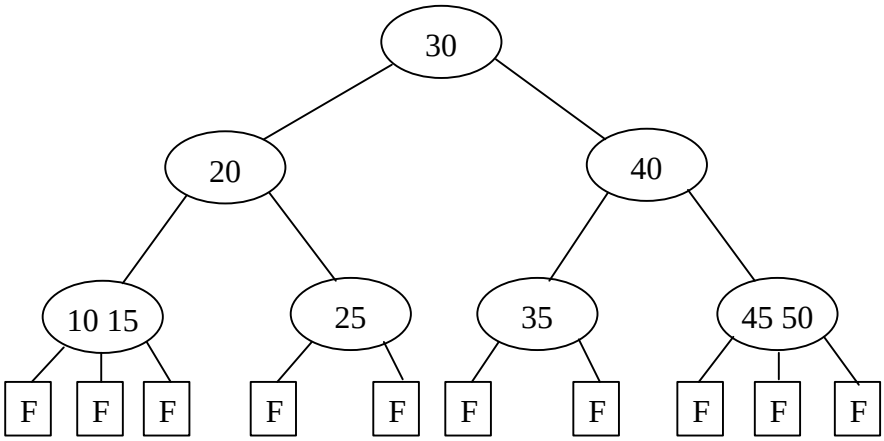
一個圖形的生成樹中，邊的加權值的最小一棵，即爲圖形的最小生成樹。

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：

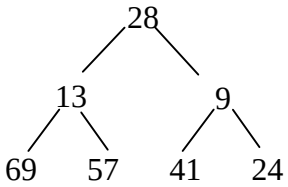


(三)Heap sort

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。

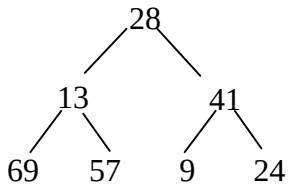
範例：一個 heap sort 的例子

原始資料：28,13,9,69,57,41,24

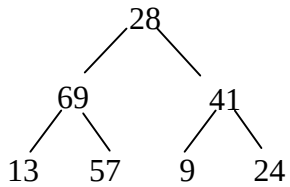


第一階段：建立堆積

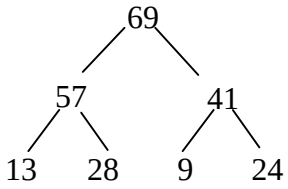
1.由 9 向下進行 sift => 28,13,41,69,57,9,24



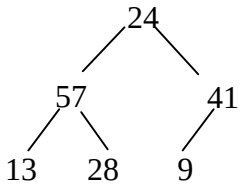
2.由 13 向下進行 sift => 28,69,41,13,57,9,24



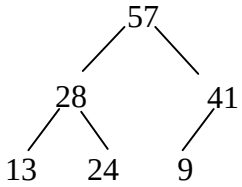
3.由 28 向下進行 sift => 69,57,41,13,28,9,24



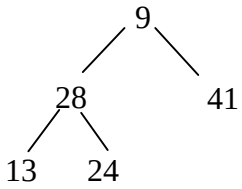
第二階段：堆積排序
1.交換 69 與 24 => 24,57,41,13,28,9,69



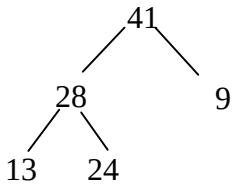
由 24(root) 向下進行 sift => 57,28,41,13,24,9,69



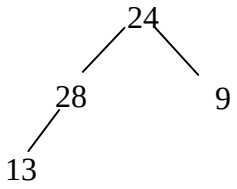
2.交換 57 與 9 => 9,28,41,13,24,57,69



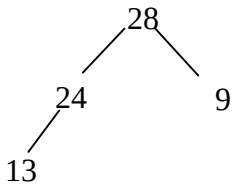
由 9(root) 向下進行 sift =>41,28,9,13,24,57,69



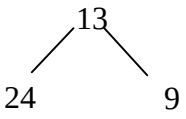
3.交換 41 與 24 => 24,28,9,13,41,57,69



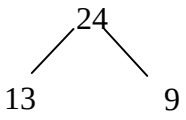
由 24(root) 向下進行 sift =>28,24,9,13,41,57,69



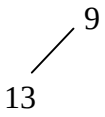
4.交換 28 與 13 => 13,24,9,28,41,57,69



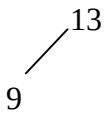
由 13(root) 向下進行 sift =>24,13,9,28,41,57,69



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



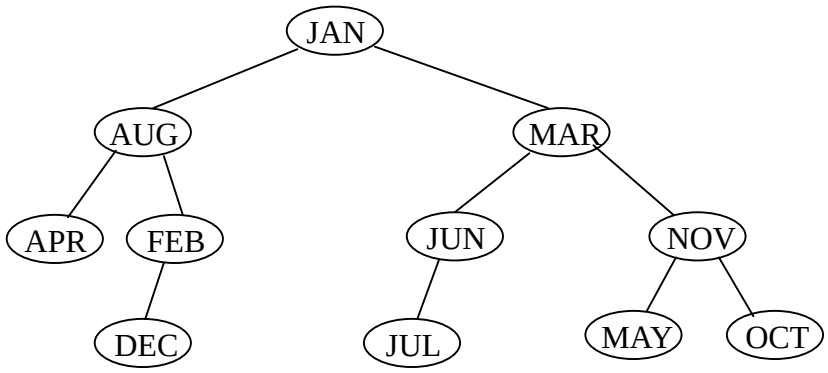
6.交換 13 與 9 =>9,13,24,28,41,57,69 (完成)

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

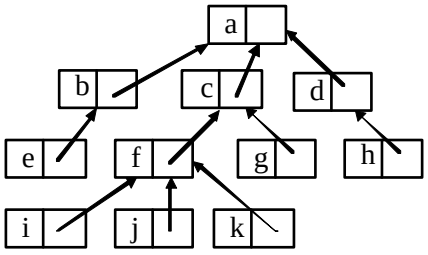


五、以下有兩種資料結構來表示一棵有根樹（rooted tree）：

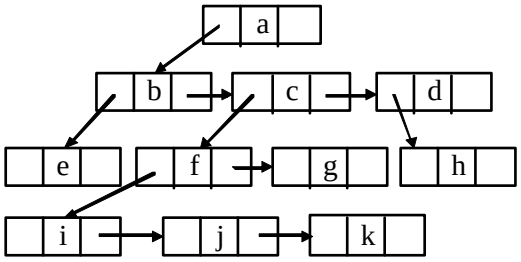
(a)每一個節點（node）有一個指標 par[ ]指向此節點的父節點（parent node）。

(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;
      }
```

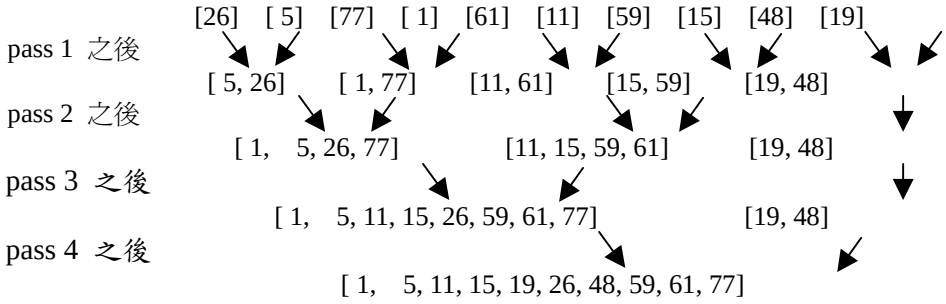
《資料結構》

試題評析	今年檢事官資料結構的命題分佈正常，五題分別測驗排序、堆積、圖形、二元樹與鏈結串列，大概比較主要的部份都有所涵蓋。不過題目難度稍微簡單了一些，因此考生要拿到高分並不難。 第一題測驗是否知道合併排序的兩種方法，反覆合併是基本的方法，應當要得到分數；遞迴法則看考生是否有準備到。 第二題是堆積的處理，大致上應該要拿到分數。 第三題是圖形追蹤問題，也測驗了 arcs 的分類，只要準備過不難取分。 第四題是二元樹追蹤問題，也十分簡單。 第五題鏈結串列的基本操作，只要能習慣虛擬碼的表示法，應該也不難取分。 預計今年考生拿到 70 分是基本底限，程度較好的考生應可拿到 85 以上分數。

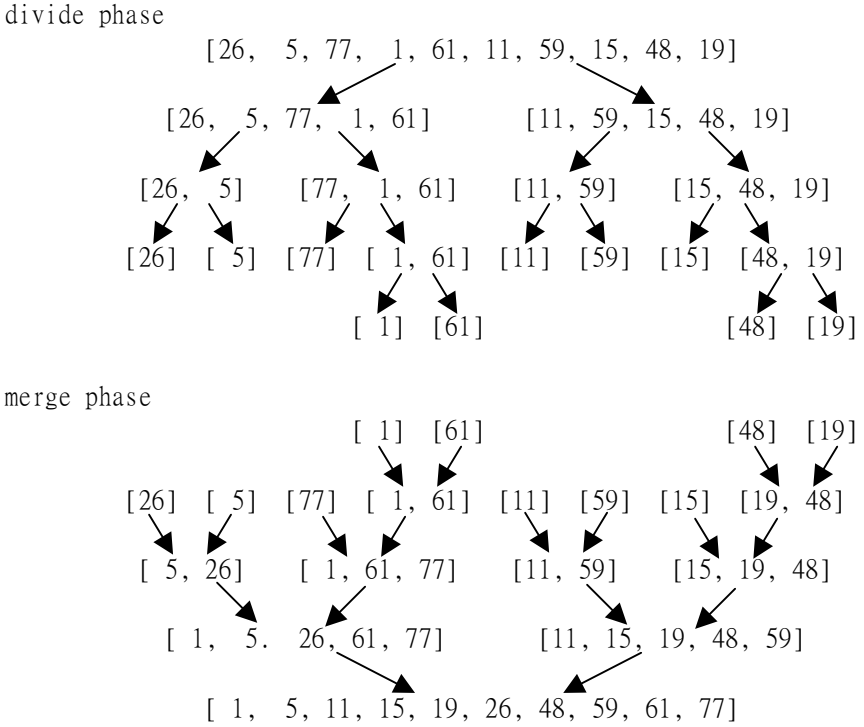
一、分別用反覆合併排序法（iterative merge sort）和遞迴合併排序法（recursive merge sort）將下列數字由小至大排序，必須列出整個排序過程。（20 分）
26, 5, 77, 1, 61, 11, 59, 15, 48, 19

【擬答】

(一)反覆合併排序法，是先將每項資料各別分成一組，再將相鄰兩組合併。



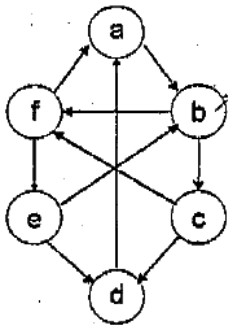
(二)遞迴合併排序法



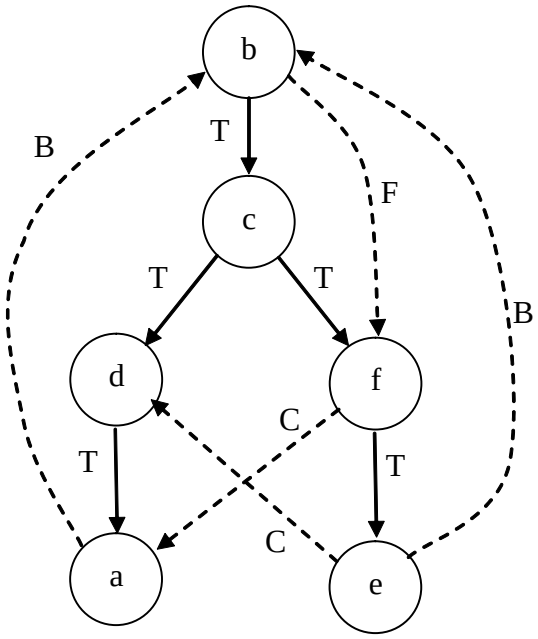
二、對陣列 A=(5, 3, 17, 10, 84, 19, 6, 22, 9)，
㈠列出堆積化（heapify）後的陣列。（10 分）
㈡接著列出加入 20 後的陣列。（5 分）
㈢再接著列出刪除最大值後的陣列。（5 分）

【擬答】
(一)堆積化之後的陣列內容爲 A=(84, 22, 19, 10, 3, 17, 6, 5, 9)
(二)加入 20 之後的陣列內容爲 A=(84, 22, 19, 10, 20, 17, 6, 5, 9, 3)
(三)刪除最大值之後的陣列內容爲 A=(22, 20, 19, 10, 3, 17, 6, 5, 9)

三、㈠依下圖建構以節點（node）b 為根（root）的深度搜尋樹（depth-first search tree），掃描邊（arc）的順序是根據字母順序。（10 分）
㈡根據建構的深度搜尋樹，列出下圖的邊那些是樹邊（tree arc）、前向邊（forward arc）、後向邊（backward arc）、交叉邊（cross arc）。（10 分）

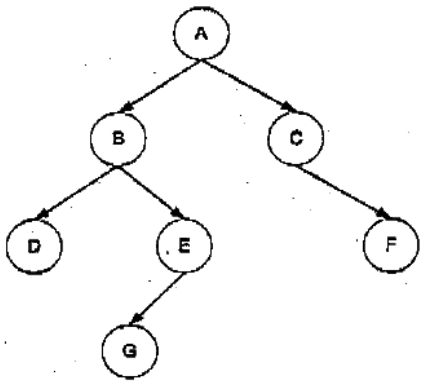


【擬答】
(一)



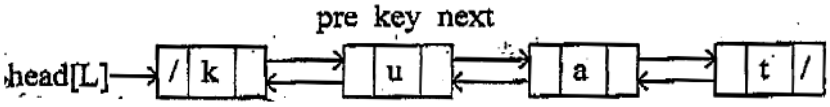
(二)tree arcs(T): <,b,c>, <c,d>, <d,a>, <c,f>, <f,e>
forward arcs(F): <b,f>
backward arcs(B): <a,b>, <e,b>
cross arcs(C):<f,a>, <e,d>

四、請將下列二元樹（binary tree）以左前序（left preorder）、右前序（right preorder）、左後序（left postorder）、右後序（right postorder）方式表示。（20 分）



【擬答】
(一)左前序(NLR)：A,B,D,E,G,C,F
(二)右前序(NRL)：A,C,F,B,E,G,D
(三)左後序(LRN)：D,G,E,B,F,C,A
(四)右後序(RLN)：F,C,G,E,D,B,A

五、在下列之雙向鏈結串列(doubly linked list)的插入及刪除副程式中，依序填入空白位置，下圖為串列之示意圖。(20 分)




```
list-insert(L,x)
{
    next[x] ← head[L]
    if head[L]≠NIL
        then _____(1)_____;
    head[L] ← x
    _____(2)_____;
}
list-delete(L,x)
{
    if prev[x]≠NIL
        then _____(3)_____;
        else head[L] ← next[x];
    if next[x]≠NIL
        then _____(4)_____;
}
```

【擬答】
(1) prev[head[L]] ← x;
(2) prev[x] ← NIL;
(3) next[prev[x]] ← next[x];
(4) prev[next[x]] ← prev[x];

《資料結構》

試題評析	今年檢事官電子資訊組資料結構考題，與先前命題型態類似，程式撰寫共佔了六十分，比例相當高，這反應了工作上的需求。不過程式撰寫題目都十分簡單，集中在堆疊處理、進位轉換與鏈結串列處理上，因此要取得分數並不難。第四題則是鄰接矩陣相關表示法，取分亦不難。最後一大題，則爲解釋名辭並要求舉例，也不難取分。估計今年考生大致取得 80 分或更高的分數並不會困難。

一、使用 C 語言，寫出以鏈結串列（Linked List）製作鏈結堆疊（Linked Stack）的副程式：void push（stack*s, float x, int *full），此副程式的功能將可以由堆疊頂端壓入一元素。（20 分）

【擬答】

```
struct node { float data; struct node *next; }
 typedef struct node * stack;
 void push(stack *s, float x, int *full)
 {    struct node * ptr;
      ptr=(struct node *)malloc(sizeof(struct node));
      if (ptr==NULL) *full=1;     // malloc 傳回 NULL 代表沒有足夠空間可以配置新節點
      else
      {    ptr->data=x;
           ptr->next=*s;
           *s=ptr;
           *full=0;
      }
 }
```

二、使用 C 語言，與遞回方式（Recursive procedure），寫出一個程式，可以將任何一個十進位數字轉換成二進位數、四進位數、六進位數與八進位數，請試著輸入十進位數字 10，並且將程式的執行結果列印出來。（20 分）

【擬答】

```
 void NumConvert(int n, int radix)
 {
     if (n>0)
     {    NumConvert(n/radix, radix);
          printf("%d", n%radix);
     }
     else printf("\n");
 }
 void main()
 {   int n;
     scanf("%d", &n);
     NumConvert(n,2);
```

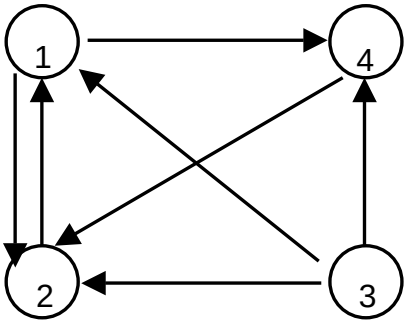
```
    NumConvert(n,4);
    NumConvert(n,5);
    NumConvert(n,6);
}
```

三、使用 C 語言，請寫出一個 ddelete （ ） 的副程式，此副程式的功能可以由含首節點之雙向環狀鏈結串列（Linked List）中刪除任意節點 P，請使用 dlist 做為指向首節點的指標。(20 分)

【擬答】

```
void ddelete(nodeptr dlist, nodeptr P)
{   if (P==dlist) printf("head node can't be deleted!\n");
    else
    {   P->left->right=P->right;
        P->right->left=P->left;
        free(P);
    }
}
```

四、在下圖中，請先使用鄰接矩陣（Adjacency Matrix）表示各個節點之間的路徑，然後再計算出各個節點之間其路徑長度為 3 的路徑數目。(20 分)



【擬答】

一、

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	1	1	0	1
4	0	1	0	0

二、任兩節點(x,y)之間，長度爲三的路徑數目如下表：

x \ y	1	2	3	4
1	1	1	0	1
2	1	1	0	0
3	2	2	0	1
4	0	1	0	1

五、請回答下列問題。(20)

(一)舉例解釋甚麼是”稀疏矩陣” (Sparse Matrix)。

(二)舉例解釋甚麼是”二分搜尋法” (Binary Search)。

(三)如果一個完滿二元樹（Full Binary Tree）的內部節點數目為 n ，請問其樹葉（leave nodes）的數目為何？

(四)舉例解釋甚麼是 “Bellman-Ford Least-Cost Multicast Tree” 演算法。

【擬答】

一、稀疏矩陣：矩陣中零元素佔大部份，只有少數元素不是零，例如：

0	0	23	0	0	0	0	0	0	0
0	0	0	0	0	18	0	0	0	0
-9	0	0	0	0	0	6	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	31	0	0	0	0	0	0	64
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	52	0	0
0	0	0	0	5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	-5	0	0	0	0	0	0	44

二、二分搜尋法：就是在一個已排序好的陣列搜尋所需資料的方法，每次在搜尋時取中間項與所需找尋的資料比較，若相等則找到；若小於中間項，則繼續搜尋中間項之左側；若大於中間項，則繼續搜尋中間項之右側。如此繼續下去，直到找到資料，或搜尋變成空的而失敗爲止。例如：在下列資料中找尋 39

13 22 39 42 58 71 76

因爲 39 比中間項 42 小，故繼續向 42 的左側搜尋，此時搜尋範圍只剩三項資料

13 22 39 42 58 71 76

因爲 39 比中間項 22 大，故繼續向 22 的右側搜尋，此時搜尋範圍只剩一項資料

13 22 39 42 58 71 76

因爲 39 等於中間項 39，故搜尋成功。

三、n+1, 因爲 Full Binary Tree 的內部節點的 degree 皆爲 2，樹葉的 degree=0，而且 $n_0=n_2+1$，故 樹葉個數 $n_0=n+1$。

四、Bellman-Ford 演算法爲 single source 的最短路徑演算法，且允許邊可以有負的成本，但不能有負的迴路(negative cycle)，如下：

```
for each vertex v in V do
begin
    if v is source then dist[v] ← 0;
    else dist[v] ← ∞;
```

```
        pred[v] ← null;
    end
    for i from 1 to |V|
        for each (u,v) in E do
            if dist[v] > dist[u] + cost[u,v] then
            begin
                dist[v] ← dist[u] + cost[u,v];
                pred[v] ← u;
            end
    for each edge (u,v) in E do
        if dist[v] > dist[u] + cost[u,v] then
            print "error: negative cycle" and exit;
```

例如：

Adjacency matrix

	A	B	C	D
A	0	2	4	-3
B	2	0	∞	3
C	4	∞	0	1
D	∞	3	1	0

Initial Dist(source=A)

	A	B	C	D
dist	0	∞	∞	∞
pred	N	N	N	N

i=1

	A	B	C	D
dist	0	2	4	-3
pred	N	A	A	A

i=2

	A	B	C	D
dist	0	0	-2	-3
pred	N	D	D	A

i=3 和 i=4 未再改變

《資料結構》

試題評析	今年檢事官資料結構的命題十分靈活，以測驗應考人員是否能活用所學過的資料結構，所考範圍皆爲常見的資料結構，主要集中在 AVL-tree、Binary Search Tree、Heap、Disjoint Set 的 trees 表示法，以及運算式的處理。雖然涵蓋範圍是一些基本的資料結構，但應考者未必習慣此類命題方式，臨場反應很重要，因此今年此一科目的分數可能會偏低，預測可能會在 40~60 之間。

一、考慮某種資源分享系統，該資源可供使用時間共分成 n 個時段，依序編號為 1、2、…、n。每一個使用者只能使用一個時段，而每一個時段最多只能有一個使用者。使用者要預約一個時段時，需透過指令 reserve(i, j)來完成，這裡 i≤j 代表使用者希望使用時段的範圍。若這個範圍的時段都已經被預約了，這指令會回傳 0 代表預約失敗，反之若這個範圍有空的時段，這指令會回傳一介於 i、j 間的整數 k，代表使用者將可以使用時段 k。

一個簡單的作法為用一個一維陣列 a[1.. n]來表示預約的狀況，a[i]=1 代表時段 i 已經被預約，a[i]＝0 代表時段 i 未被預約。實做指令 reserve(i, j)時，只需一一檢查 a[i]到 a[j]是否有值為 0 即可。因此這個指令的執行時間為 0(j－i)，在最差的狀況下這指令的執行時間可高達 0(n)。事實上有在最差狀況下執行時間為 0(log n)或更好的作法。

請(一)設計一個能表示整個預約狀況的資料結構，與(二)寫出一實做 reserve(i, j)的演算法，使得在最差情況下 reserve(i, j)的執行時間為 0(log n)或更好。(20 分)

【擬答】

一、使用 AVL-tree 來記錄尚未被預約的時段，將每一個未被預約的時段 k，分別建立一個節點 k，並將其記錄在 AVL-tree 中。一開始時，因爲時段 1~n 都未被預約，故建立 n 個節點分別記錄 1~n，並建立起一棵 AVL-tree。

要進行 reserve(i,j) 時，只要在 AVL-tree 中找到一個節點 k(i≦k≦j)，就表示有符合要求的時段被找到，然後將節點 k 自 AVL-tree 中刪除即可。

(一)

```
int reserve(treeptr tree, int i, int j)
{    if (tree==NULL) return 0;
     else if (tree->timeperiod < i) return reserve(tree->rightchild, i, j);
     else if (tree->timeperiod > j) return reserve(tree->leftchild, i, j);
     else
     {     int t=tree->timeperiod;
           deleteAVLtree(tree);
           return t;
     }
}
```

二、二元搜尋樹(binary search tree)是一種基本的資料結構，在實做上有時需將兩個二元搜尋樹 T_1、T_2 合併成一個新的二元搜尋樹，一個單純的作法是：先建一個新的且為空的二元搜尋樹 T，然後將 T_1、T_2 裡的數字一個一個依任意次序拿出，再插入(insert)到 T 裡面。這個作法的缺點是，在最差的情況下，運算時間會高達 $O(n^2)$，其中 n 為 T 的大小。實際上有一個運算時間為 O(n)的作法：

(一)請描述一個線性時間的作法，可以將一個二元搜尋樹裡的數字依小到大的次序排出來，並放在一個一維陣列上。(5 分)

(二)利用(一)的特性，請設計出一個線性時間的演算法將 T_1、T_2 裡的數字依小到大的次序排出來，並放在一個一維陣列上。(5 分)

(三)給 n 個已經排好的數，請設計出一個線性時間的演算法建構出以這 n 個數為鍵值(key)的二元搜尋樹，同時我們要求這個樹的高度必須為 O(log n)。(10 分)

【擬答】

一、使用二元樹的中序追踪法，來追踪二元搜尋樹，其追踪的次序正是由小而大的順序；在追踪時，每次追踪到一個節點，就將節點資料依序置入一維陣列即可。因爲二元樹的中序追踪所需的時間，與節點個數成正比，故時間爲線性時間。

二、分別對兩棵二元樹進行中序追踪，得到兩組由小而大的序列，再將這兩個序列合併。若兩棵二元搜尋樹分別有 n1 與 n2 個節點，則中序追踪時間爲 O(n1)與 O(n2)，合併所需時間爲 O(n1+n2)，因爲 n=n1+n2，故總時間爲 O(n)。

三、使用 divide-and-conquer 方式，取中間項做爲樹根，中間項之前的資料以遞迴的方式建立成左子樹；中間項之後的資料以遞迴的方式建立成右子樹。

```
treeptr CreateTree(int a[], int low, int high)
{       if (low>high) return NULL;
        else
        {   int mid=(low+high)/2;
            treeptr p=(treeptr)malloc(sizeof(node));
            p->data=a[mid];
            p->leftchild=CreateTree(a, low, mid-1);
            p->rightchild=CreateTree(a, mid+1, high);
            return p;
        }
}
```

三、一個監測器(sensor)可看成是一台計算機，只是其中央處理器速度比一般計算機慢，且記憶體也較小。有一監測器每隔一段時間需從外界讀入一個監測值，一段時間下來總共讀入 n 個數值，而這監測器的任務為記性這 n 個數裡面最小的 k 個，一般來講 k 遠小於 n。現在監測器的記憶體剛好只能放 k 個數值，外加一個暫存器存放剛從外界讀入的監測值。同時由於監測器的處理速度不快，根據估計，若在監測器讀取兩個監測值的間隔時間內，處理器需做 k/2 次以上的比較，那麼監測器可能會錯失一個監測值，長期下來會造成紀錄結果不正確。這裡我們假設 k≥100。請設計出一個可行的作法，可以讓這個監測器正確的完成其任務。(20 分)

【擬答】

(1)使用最大堆積(max-heap)記錄最小的 k 個值。

(2)每次讀入一個值後，與 root 比較，若新讀入的值大於 root，則將新讀入值捨棄。

(3)若新讀入之值小於 root，則以新讀入之值取代掉原來的 root 的值，並且由 root 向下進行調整，使其回復爲最大堆積之情況。

(4)按此方法，每次讀入一個值，最多只需要做 $2\lfloor \log_2 k \rfloor$ 次比較。

四、假設有一個數學運算式(expression)裡面每一個運算元(operand)都是一個正整數，而運算子(operator)只有加法與乘法兩種。請就以下每一數學運算式的格式限制，寫出一段程式算出這個數學運算式的值。(20 分)

(一)數學運算式為前序表示(prefix expression)

(二)數學運算式為後序表示(postfix expression)

這裡你可以假設已經有一個前處理程式 get-next-token()可以運用，其規格為：同傳值為正整數，表示 get-next-token()讀到一個運算元，其大小即為回傳值。

回傳值為 0，表示 get-next-token()讀到運算式的結尾。

回傳值為－1，表示 get-next-token()讀到加號。

回傳值為－2，表示 get-next-token()讀到乘號。

【擬答】

一、

```
int PrefixEval()
{   int x=get_next_token();
    if (x==0) Error();
    else if (x==-1) return PrefixEval()+PrefixEval();
```

```
        else if (x==-2) return PrefixEval()*PrefixEval();
        else return x;
    }
```

(一)
```
int PostfixEval()
{       int x;
        InitStack();
        x=get_next_token();
        while (x!=0)
        {
            if (x>0) Push(x);
            else if (x==-1)
            {   int p=Pop();
                Push(Pop()+p);
            }
            else if (x==-2)
            {   int p=Pop();
                Push(Pop()*p);
            }
            x=get_next_token();
        }
        return Pop();
}
```

五、一個軟體系統內的程式，常需檢查其輸入資料格式是否正確，以確保後續處理不出問題。現有一組輸入資料放在一個陣列 p[1.. n]上代表一棵有根樹(rooted tree)，其中 n 為節點數，且節點編號為 1, 2, …, n。若 p[i]存的值為 j，表示節點 i 的父節點為 j。若節點 i 為根節點則 p[i]存的值為 i。請寫出一個線性時間的演算法檢查這陣列是否表示一棵有根樹。(20 分)

【擬答】

一、使用 disjoint set 的運算 find(i)，並用 path compression 來將所有節點直接指向其 root，最後再檢查所有的節點是否具有相同的 root。 演算法如下：

```
for i ← 1 to n do
  begin
    k←i; count←0;
    while p[k]≠k and count≦n do
      begin
          count←count+1;
          k←p[k];
      end;
    if count>n then Error();
    else
      begin
```

```
            root←k;
            k←i;
            while p[k]≠root do
               begin
                  m←k;
                  k←p[k];
                  p[m]←root;
               end;
         end;
   end;
if p[1]~p[n] are identical then O.K.
else Error();
```


《資料結構》

試題評析	今年檢事官資料結構的考題相當簡單，主要都在測驗應試者的觀念，沒有複雜及刁鑽的問題。第一題是測驗應試者是否了解資料結構、演算法與程式之間的關聯性。第二題則是基本的運算式換的問題，大部份著重在觀念較多。第三題是幾個基本線性資料結構的觀念題，取分不難。第四題必須先了解題目的需求，只是要檢查是否存在到每一個頂點的路徑，故只需要使用 DFS 或 BFS 即可，不需要使用像 Dijkstra 一類複雜的演算法。今年應試者在資料結構一科中可拿到不錯的分數，分數應可取得 60 分以上，程度較佳者要拿到 80 分的成績應有可為。

一、請論述 Data Structures + Algorithms = Programs 之意義。 (二十五分)

【擬答】

當要撰寫一個程式時，在了解到問題的要求之後，通常必須先設計所需要的演算法，然後再依演算法所需要的運算，選擇最適當的資料結構，來減少運算所需要的時間，然後評估其時間效能是否符合要求；如果不符合需求，就必須嘗試修改演算法或更改使用的資料結構，看看是否可以將執行時間再降低。

在確認演算法與資料結構之後，可以再選擇適當的程式語言，以便將程式實際撰寫出來。程式撰寫時，先定義資料結構所要儲存的資料的儲存結構，例如陣列、鏈結串列的節點結構，或所需要的變數等等；再寫輸入資料的程序，將輸入資料以預先選好的資料結構形式儲存下來；然後撰寫處理的程序，處理程序通常就是根據規劃好的演算法步驟，一步步寫出處理的程式命令；最後撰寫輸出結果的程序，而將程式完成。

因此，撰寫程式前，所需要的準備工作，就是要先規劃好演算法，使解決問題的過程以明確的步驟呈現出來；而資料結構的規劃，則是牽涉到資料在電腦的記憶體中的儲存方式，如何能協助演算法在最短的時間內，完成其每個步驟的功能。

二、(一)在什麼狀況需要把 infix expression，例如 a + b 改成 postfix expression ab +？(十分)

(二)轉換時要用那一資料結構來做轉換。(五分)

(三)請敘述轉換的方法？(十分)

【擬答】

(一)當電腦要計算算術式的結果時，因為 infix expression 的計算方式較為複雜，所以通常會先將 infix expression 先轉換成為 postfix expression，再加以計算，這樣會比較好處理。此外，compiler 中也常將 infix expression 先轉換成為 postfix 的中間碼，然後根據 postfix expression 翻譯成最後的機器目的碼。另外還有一種情形，就是計算運算式的機器結構是 stack machine，此時也必須將 infix expression 先轉換成為 postfix 形式再計算。

(二)infix expression 轉換為 postfix 形式時，需要使用 stack 結構，因為 postfix expression 的計算是由前而後掃瞄 postfix expression，先遇到的 operator 會先計算。因此，infix expression 中 priority 較低的 operator，通常會先被置於 stack 中，在較後的階段才輸出到 postfix expression，以配合 postfix expression 的 evaluation rule。

(三)infix expression 轉換成 postfix expression 的方法如下：

1.由前而後逐一掃瞄 infix expression 的 element。

2.若 element 為 operand，則直接輸出。

3.若 element 為 operator，則必須與 stack top 的 operator 比較其計算的先後次序。

如果 element 須優先計算，則將 element push 到 stack 中；

如果 stack top 的 operator 須先計算時，則將 stack top 的 operator 先取出並且輸出，然後重新做步驟 3。

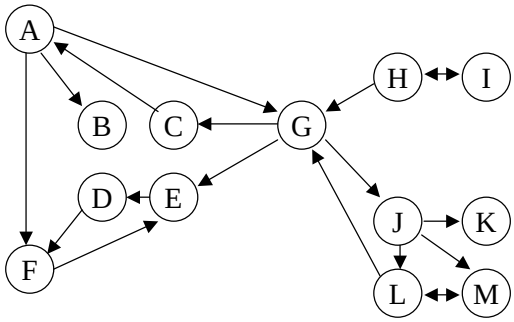
4.當掃瞄完 infix expression 後，若 stack 仍中還積存有 operator(s)，則必須將這些 operator(s) 逐一取出並且輸出。

三、請比較 array，list 及 queue 的使用時機，並舉例說明其優缺點。(二十五分)

【擬答】

	使用時機	優點的例子	缺點的例子
Array	需要對其中的資料項目，進行直接存取時，使用陣列會較爲適合。	二分搜尋法、雜湊表。	不適合經常 insert/delete 的資料
List	經常需要插入新資料或刪除舊資料的動態資料，比較適合使用 linked list 來儲存，可以減少所需要的資料搬動次數。	經常要 insert/delete 的一群資料	不適合做二分搜尋
Queue	適合於依照 FIFO 方式處理資料時，適合使用 queue。	FIFO scheduling	不適合做資料次序的反轉動作

四、給予一有向圖（directed graph）如下： (二十五分)



若存在一路徑可以從 x 到 y，且存在一路徑可以從 y 到 z，則必可找到一個路徑從 x 到 z。以上圖來看，有一個路徑可以從 A 到 G，並有一個路徑可以從 G 到 E，則，我們可以找到一個路徑從 A 到 E。但從上圖中，我們卻無法找到一個路徑從 A 到 I。請問，該以何種資料結構來幫助你找到所有的路徑？如何做？意即從圖中的任一點出發，是否存在路徑可以到達圖中之其他點。

【擬答】

可以使用 DFS(Depth-First Search)或 BFS(Breadth-First Search)來找路徑。以下是以 DFS 來處理。

```
push (起點);
while stack not empty do
 begin
     u <- pop();
     If   u is not visited then
       begin
           mark u is reachable;
           for each v adjacent from u do
                if v is not visited then Push(v);
       end;
 end;
```

資料結構：

(1)使用 adjacency list 來記錄圖形的 vertices 與 edges 的關係。

(2)使用 stack 來記錄可以到達的 vertices。

《資料結構》

試題評析	本次檢察事務官之資料結構試題，比前幾次要來得不易作答，主要是應用變化題與程式撰寫題居多。即使應試者每題都能夠迅速掌握到要點，在時間限制之內要完全作答，恐亦非易事。但也因而更能測驗出應試者，對相關知識的熟悉程度與應變能力。此外，因為題目份量多，有些又具有難度，應試者的情緒，能否冷靜作答，可能反而是影響成績的重大因素。大致而言，若不考慮答題時間是否充足之問題，可算是相當不錯的命題。 各題的特點如下：第一題是鏈結串列的應用題，屬於程式設計基礎問題，主要在測驗應試者的鏈結串列處理能力與長整數加法的基本計算法。第二題為較基本的問題，二元搜尋樹基本操作與追蹤法，對應考者應該可以輕易拿分。第三題為二維陣列空間安排的變化題，難度不算難，但須冷靜作答，應可以拿到分數。第四題必須先想到要使用基數排序法，否則無法做到時間複雜度O(N)的要求，但仍須撰寫出程式，最重要的還是要「冷靜作答」。第五題為鏈結串列基本處理，要拿分應不難。第六題須先做遞迴關係式的分析，但後半段非遞迴程式撰寫部份，須使用動態程式設計法，否則較難寫出來。 整份試題除了第二題與第五題較基本之外，其餘問題，幾乎題題皆關鍵，多答對任何一題，都會為自己增加不少的上榜機會。預估本科目應考者的分數範圍應分佈在40~80之間。

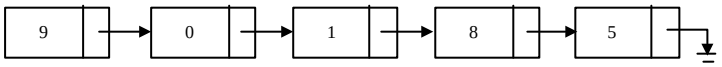
一、設計一種資料結構可以用來儲存任意大小的非負整數（包含零及正整數），並請寫一段程式在這資料結構上做兩非負整數的加法運算。（15分）

【擬答】

使用鏈結串列來儲存此一非負整數，其節點結構如下：

```
struct node
{    int digit;
     struct node * next;
};
typedef struct node * nodeptr;
```

例如：58109 可以建立如下之之鏈結串列，個位數字在最開頭，愈高位數字儲存在串列愈尾部



兩串列數字法運算如下：

```
nodeptr listadd (nodeptr p, nodeptr q)
{  nodeptr head, tail, r;
   int pdigit, qdigit;
   int carry=0, sum;
    head=(nodeptr)malloc(sizeof(struct node));
    tail=head;
   while (p!=NULL && q!=NULL)
   {  if (p==NULL) pdigit=0;
      else {    pdigit=p→digit;
                p=p->next;        }
     if (q==NULL) qdigit=0;
      else {    qdigit=q->digit;
                q=q->next;        }
       sum=pdigit+qdigit+carry;
       tail->next=(nodeptr)malloc(sizeof(struct node));
       tail=tail->next;
       tail->digit=sum%10;
       carry=sum/10;
    }
    tail->next=NULL;
    r=head->next;
    free(head);
   return r;
 }
```

二、假設一組數為：A=8，B=17，C=15，D=10，E=12，F=19，G=6，H=30，I=5，J=14**考慮以下序列運算：**

Insert(A)，Insert(B)，Insert(C)，Insert(D)，Insert(E)，Insert(F)，Insert(G)，Insert(H)，Insert(I)，Insert(J)，

Delete(A)。（20分）

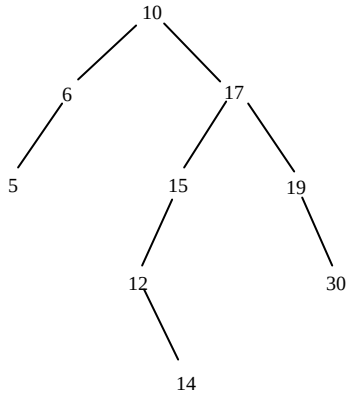
(一)請描述binary search tree的特性，並請畫出經過以上序列運算所建構的binary search tree。

(二)請寫出所建構之binary search tree的preorder，inorder，postorder，breadth-first order。

【擬答】

(一) Binary Tree 的特性：是一棵 binary tree，若其不是空的 binary tree，則樹根(root)的左子樹中的節點所儲存之資料，皆不大於樹根；而右子樹中的節點所儲存的資料，皆不小於樹根。而且、右子樹亦皆為 binary trees。

這組數所建立之最後 binary tree 如下：



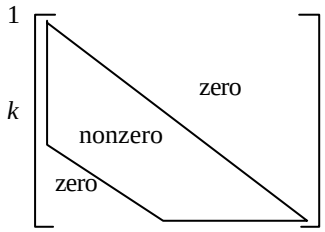
(二) preorder: 10,6,5,17,15,12,14,19,30
inorder:5,6,10,12,14,15,17,19,30
postorder:5,6,14,12,15,30,19,17,10
breadth-first order:10,6,17,5,15,19,12,30,14

三、一個n乘n的方陣A=[ai,j]，1≤i≤n，1≤j≤n，若存在一數k，k≤n，使得i–j≥k與i–j<0時ai,j=0；則此方陣稱為下三角斜條方陣（lower triangular banded matrix）。若0≤ i–j<k則稱元素ai,j，在斜條（band）上。

請設計一方法將斜條上的元素儲存至一維陣列D。（20分）

(一)問D總共有多少元素。

(二)假設0≤ i–j<k，請寫出ai,j儲存於陣列D的指標位置。



【擬答】

(一) 共有 $n+(n-1)+(n-2)+...+(n-k+1)=\frac{(2n-k+1)k}{2}=nk-\frac{k(k-1)}{2}$ 個元素。

(二) 以 row-major 方式來安排 non-zero 元素的空間即可，a11 置於 D[1]，a21 置於 D[2]，a22 置於 D[3]，a31 置於 D[4]，餘此類推。若 aij 置於 D[k]，則 k 與 i,j 之關係式如下:

$$k=\begin{cases}\frac{i(i-1)}{2} & ,1\le i\le k\\ \frac{k(k+1)}{2}+k(i-k)-i+j & ,k<i\le n\end{cases}$$

註：此題亦可採用對角線的次序，由最下方(或最上方)的對角線開始，一條一條向上(向下)逐一安排空間，則公式將有所不同。

四、假設有一組整數，每個數都介於0與31之間。請寫一個程式將這組數依小到大排列。若這組整數的個數為*N*，這程式所需要的計算次數最多為O(*N*)。（20分）

【擬答】

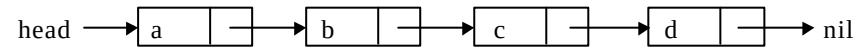
使用 radix=2 的 LSD(Least-Significant Digit) Bucket Sort 即可，假設N 個資料已建立並儲存於 single linked list 中，兩個 buckets(以linked lists表示) 皆各有首節點，並以 head[0],head[1]與tail[0],tail[1]分別指向開頭與結尾的節點。

```
nodeptr LSDBucketSort(nodeptr p)
{   nodeptr head[2], tail[2];
    int pass, m;
    nodeptr q;
    head[0] =(nodeptr)malloc(sizeof(struct node));
    head[1]=(nodeptr)malloc(sizeof(struct node));
   for (pass=1, m=1; pass<=5; pass++, m*=2)
    {   tail[0]=head[0];   tail[1]=head[1];
        while (p!=NULL)
        {   q=p;   p=p->next;
            i=(q->data & m) >> (pass-1);
            tail[i]->next=q;
            tail[i]=q;
        }
        tail[0]->next = tail[1]->next = NULL;
        if (head[0]==tail[0]) p=head[1]->next;
        else { p=head[0]->next; tail[0]->next=head[1]->next; }
    }
    return p;
}
```

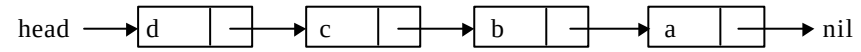
因為總共進行 5 passes 的運算，而每一回所需的時間皆為 O(N)，故總時間仍為 O(N)。

五、寫一程式將一個linked list的所有元素（entries）的次序完全倒過來（如下圖所示）。（10分）

原來的linked list：



次序完全倒過來的linked list：



【擬答】

```
nodeptr ListReverse(nodeptr head)
{   nodeptr reversed, temp;
    reversed=NULL;
    while (head!=NULL)
    {   temp=head;
        head=head->next;
        temp->next=reversed;
        reversed=temp;
    }
   return reversed;
}
```

六、令長度為n的序列集合

An={x1,x2,…,xn | xi=0,1**或**2，**對**1≤i≤n−1，xi,xi+1≠1,1**且**xi,xi+1≠2,2}（**即序列中沒有連續兩個1或是連續兩個2**）。

設f(n)**為**An**的個數，**f0(n), f1(n), f2(n)**分別為**An**中**x1=0,1,2**的序列的個數。請導出**f(n)**的遞迴公式（**recursive formula**），並請寫一非遞迴函式計算**f(n)。（15分）

【擬答】

遞迴公式為

$$f_k(n)=\begin{cases} f_0(n-1)+f_1(n-1)+f_2(n-1) & ,n>1\text{且}k=0 \\ f_0(n-1)+f_2(n-1) & ,n>1\text{且}k=1 \\ f_0(n-1)+f_1(n-1) & ,n>1\text{且}k=2 \\ 1 & ,n\le 1 \end{cases}$$

而 $f(n)=f_0(n)+f_1(n)+f_2(n)$

故可以採用 dynamic programming **方式來計算** $f_k(n)$**，並以一個** 2-D array a[k][m] **來儲存計算過的**$f_k(m)$**，其中**$1\leqq m\leqq n$**，**$0\leqq k\leqq 2$**程式如下：**

```
int f(int k, int n)
 {   int m;
     int a[3][MaxN];
      a[0][1]=a[1][1]=a[2][1]=1;
     for(m=2; m<=n; m++)
      {   a[0][m]=a[0][m-1]+a[1][m-1]+a[2][m-1];
          a[1][m]=a[0][m-1]+a[2][m-1];
          a[2][m]=a[0][m-1]+a[1][m-1];
      }
     return a[0][n]+a[1][n]+a[2][n];
 }
```