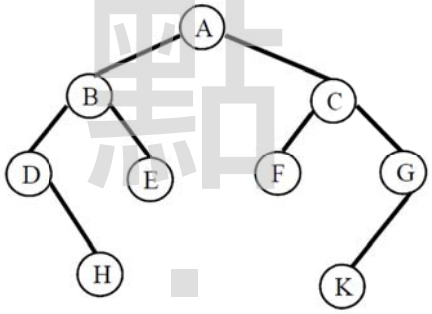
《資料結構》

一、(一)請分別寫出下圖二元樹的前序走訪法（preorder traversal）、中序走訪法（inorder traversal）、後序走訪法（postorder traversal）的結果（6分）

(二)請在無法預知二元樹的節點數條件下，設計在程式中表示二元樹的資料結構。再假設二元樹已依前述結構儲存在程式，設計一副程式（或函式）的演算法，在提供樹根給此副程式（或函式）後，其執行二元樹中序走訪法的程序並輸出走訪結果。此副程式（或函式）不可使用遞迴呼叫技術但可添加其他資料結構，演算法的時間複雜度和空間複雜度須均為O(n)，n為二元樹的節點個數。演算法可以虛擬碼（pseudo-code）或以高階語言如C呈現。需分析說明副程式（或函式）演算法的時間複雜度和空間複雜度均為O(n)。（提醒：若用遞迴呼叫技術設計，演算法部分不給分）（13分）

(三)請分別說明在程式執行過程，以第(二)子題非遞迴呼叫技術設計相較於以遞迴呼叫技術設計在時間與空間的效能優勢各為何？（6分）



試題評析	本題重點在測驗不以遞迴方式，進行二元樹的追踪方法，可以使用堆疊來記錄所有尚未追踪的方份，再逐一取出，予以追踪即可。
考點命中	1.《資料結構》，高點文化出版，王致強編著，頁6-16 精選例題14。 2.《資料結構》，高點文化出版，王致強編著，頁6-20 精選例題18。

答：

(一) 前序走訪法：ABDHECFGK
中序走訪法：DHBEAFCKG
後序走訪法：HDEBFKGCA

(二) 使用stack來記錄binary tree中，尚未追踪的部份。
每次push到stack的是一對資料(pointer,flag)，flag=False時，pointer是一棵尚未追踪的subtree；flag=true時，pointer是尚未追踪的單一節點。
inorder traversal 程序如下：

```
1.  void preorder(treeptr t)
2.  {   Boolean flag;
3.      treeptr p;
4.      Stack_Init();
5.      Stack_Push(t, False);
6.      while (!Stack_Empty())
7.      {
8.          Stack_Pop(p,flag);
9.          if (p!=NULL)
10.             if (flag) output(p->data);
11.             else
```

```
12.             {
13.                 Stack_Push(p->right,False);
14.                 Stack_Push(p, True);
15.                 Stack_Push(p->left,False);
16.             }
17.     }
18. }
```

(三) 非遞迴呼叫技術設計相較於以遞迴呼叫技術設計
時間：兩者時間複雜度相同，皆為O(n)。
空間：非遞迴使用的堆疊如果是鏈結串列實作，空間複雜度=O(tree height)；遞迴呼叫空間與最深的呼叫層次成正比，所以同樣也是O(tree height)。

二、二維方陣A大小為n×n，方陣中的元素除了主對角線之元素以及緊鄰它的上下兩條對角線之元素的值可能不為零外，方陣A其他元素之值一定為零，以5×5方陣為例如下圖。請以一維陣列B設計儲存此方陣A之結構，陣列B之索引值自0開始，且陣列B的元素數量須小於或等於3×n-2。設計的結構須包含如何有效率地決定儲存方陣A之元素a_{ij}以及如何自陣列B中取得或決定方陣中元素a_{ij}值，其中$0 \le i, j \le n-1$ 而i與j分別為元素在方陣A中之列號與行號。（20分）

$$\begin{bmatrix} a_{00} & a_{01} & 0 & 0 & 0 \\ a_{10} & a_{11} & a_{12} & 0 & 0 \\ 0 & a_{21} & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{43} & a_{44} \end{bmatrix}$$

試題評析	本題屬於稀疏矩陣的表示技巧，目的在節省空間及存取效率的問題考量。
考點命中	《資料結構》，高點文化出版，王致強編著，頁2-17 精選例題20。

答：
將tri-diagonal matrix的非零元素，以row-major order存於1-D array B[0..3n-3]。
Location Function: Loc(a_{ij})=3*i+j-i=2*i+j
可以使用下面函數來儲存aij的值value：

```
storeA(int i, int j,int value) {
    int k=2*i+j;
    B[k]=value;
}
```

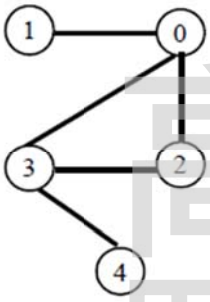
可以使用下面函數來取得aij的值：

```
fetchA(int i, int j) {
    int k=2*i+j;
    return B[k];
}
```

三、(一)請畫出下圖以鏈結串列（link list）為基礎的相鄰串列（adjacency list）結構表示之結果。（5分）
(二)請運用一維陣列設計一資料結構採循序串列（sequential list）架構，其仍舊以類似子題(一)相鄰串列策略表示無向圖(undirected graph)節點與邊的關係，但僅以一維陣列呈現第(一)子題之相鄰串列概念。圖之節點與邊的關係僅以此一維陣列元素記錄並

呈現，不可使用其他資料結構，另外，陣列中亦需記錄此陣列中用來記錄與圖相關資訊之元素個數；除了說明資料結構外，也請寫出下圖以此資料結構表示之一維陣列結果。（8分）

(三)請列出兩項在程式中以第(一)子題之以鏈結串列（link list）表示圖比以第(二)子題一維陣列表示圖適合的應用情境或效能優勢。另外，也請列出兩項在程式中以第(二)子題一維陣列表示圖比以第(一)子題鏈結串列（link list）表示圖適合的應用情境或效能優勢。（12 分）

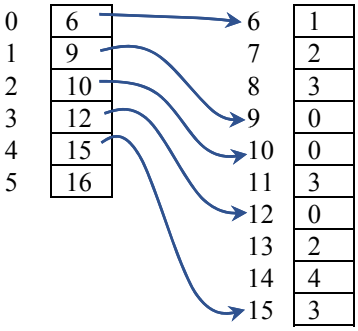


試題評析	本題是圖形表示法問題，與表示法優缺點的比較。
考點命中	《資料結構》，高點文化出版，王致強編著，頁8-9。

答：

(一) Adjacency List 如下：

(二) Sequential List 如下：



(三) 第(一)子題之以鏈結串列的效能優勢：

(1) 適合儲存動態圖形，可以方便調整圖形的結構

(2) 空間利用率較高，不需要事先配置陣列的空間，不會浪費空間。

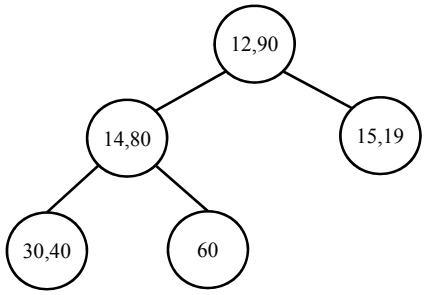
第(二)子題之以鏈結串列的效能優勢：

(1)適合儲存靜態圖形，陣列存取時不用讀取鏈結指標，速度會優於鏈結串列。
(2)計算一個節點的 degree，時間很快，為 O(1)。

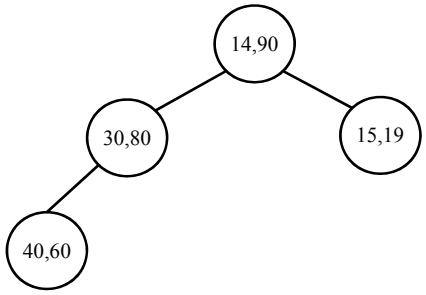
四、區間堆積（interval heap）是一種優先佇列（priority queue），請回答下列相關的問題。
(一)從一個沒有元素的區間堆積開始，依序插入40, 30, 60, 15, 14, 19, 80, 12, 90等元素。請畫出最後區間堆積的樹狀結構圖。（9分）
(二)請自第(一)子題建構的區間堆積中刪除元素12，並畫出刪除該元素後區間堆積的樹狀結構圖。（3分）
(三)請以一維陣列設計資料結構儲存區間堆積，該資料結構可以透過節點對應之陣列索引值index構成的數學式計算出其父節點parent、左子節點left、右子節點right與兄弟節點brother等在陣列中的索引值。假設此一維陣列之起始索引值為0，請列出由index構成的計算parent、left、right、brother的數學式。並請畫出以此一維陣列儲存第(一)子題建構完成的區間堆積的結果。（12分）
(四)舉例並說明一既需要提供最高優先元素，也需要提供最低優先元素的優先佇列的應用實例或系統。（6分）

試題評析	本題是區間堆積的運算操作，應用實例及二元樹的Implicit array表示法。
考點命中	1.《資料結構》，高點文化出版，王致強編著，頁7-18~7-26。 2.《資料結構》，高點文化出版，王致強編著，頁7-3~7-4。

答：
(一)



(二)delete-min(12)之後



(三)Tree 的 Implicit Array 表示法, 假設 root 的索引為 1
node 的 索引為 index
(1) parent 的索引為 $\left\lfloor \frac{i}{2} \right\rfloor$。
(2) left child 的索引為 2i。

(3) right child 的索引為 2i+1。

(4) brother 的索引為 $\begin{cases} i+1, & i\text{為偶數} \\ i-1, & i\text{為奇數} \end{cases}$

第(一)子題的表示

0	1	2	3	4	5	6
-	(12,90)	(14,80)	(15,19)	(30,40)	(60,-)	-

(四) printer 列印緩衝區管理

(1) 當有工作要列印時，將列印的工作 Insert 加入 Interval Heap。

(2) 每件列印工作都有 proirity，priority 值愈大愈優先，可以使用 extract-max 取出優先等級最高的工作列印。

(3) 當緩衝空間不足時，可使用 extract-min 取出並放棄優先等級最低的工作。