《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);
```

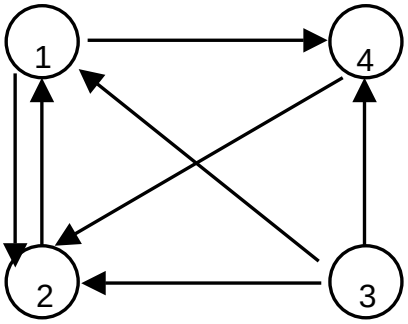
```
    NumConvert(n,4);
    NumConvert(n,5);
    NumConvert(n,6);
}
```

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