淡江大學 100 學年度碩士班招生考試試題

92-1

系別：資訊管理學系　　科目：資　料　結　構

考試日期：2 月 28 日(星期一) 第 3 節　　本試題共　8　大題，　4　頁

本試題雙面印刷

1. 請完成以下 Linked List 的 Java 實作，爲了簡化實作，List 的最前端被加入一個空的標頭節點 (dummy head node)。(10%)

```
class Node {
    Object obj;     Node next ;
    Node() { }
    Node(Object o,Node n) { obj=o; next=n;}
}
class LinkedList {
    Node head = new Node() ;
    // 將物件 o 插入成爲第 index 個元素, index 由 0 起算
    void insert(int index, Object o) {
        Node p = head ;
        for (int i = 0; ____❶____; i++) p=p.next ;
        p.next = new ____❷____ ;
    }
    void remove(Object o) { // 刪除串列中所有的物件 o
        Node p = head ;
        while (____❸____) {
            if (p.next.obj.equals(o))
                ____❹____;
            else
                ____❺____;
        }
    }
}
```

2. Stack 與其應用: (10%)

(a) 寫出以下運算式的後置式(postfix expression)。(3%)

(-a+b*c)-d-e*f/(-g/h)/i

(b) Stack 之 pop()經常以如下方式來實作，在實際執行時，會產生何種記憶體管理問題? (4%)

```
class Stack { // Java 實作
    Object[] data=new Object[MAX_SIZE] ;
    int top = -1;
    …
    Object pop() {
        if (!empty()) return data[top--] ;
        return null ;
    }
}
```

(c) 若改用 linked list 來實作 Stack，有何優點與缺點? 請依照優、缺點分項條列，否則不給分。(3%)

背面尚有試題

92-2

3. 請完成以下 circular queue 的 Java 實作。(10%)

```
class CircularQueue {
    int[] queue = new int[6];
    int front=0, rear=__❶__;   //front, rear 分別記錄隊伍第一個與最後一個元素的註標
    int cnt=0; //記錄隊伍中的元素個數

    public int size() { return queue.length; }
    public void enqueue(int x) {
        if (cnt != size()) {
            ____❷____ ;
            queue[rear] = x ;
            cnt++ ;
        }
    }
    public int dequeue() {
        if (cnt==0)
            return Integer.MAX_VALUE ;
        else {
            int buf =____❸____;
            ____❹____;
            cnt-- ;
            return buf ;
        }
    }
    public void print() {   //依序印出 queue 的內容
        for (int i = 0; i<cnt; i++){
            System.out.print(queue[____❺____]+" ") ;
        System.out.println();
    }
}
```

4. Binary Trees: (10%)

(a) 依序將以下鍵值加入一棵空的二元搜尋樹，以圖形表示你的答案。(3%)

12, 10, 6, 20, 14, 22, 24, 8, 23

(b) 說明引線二元樹(threaded binary tree)的用途。(3%)

(c) 繪製(a)對應之引線二元樹(threaded binary tree)。(4%)

5. AVL Trees: (12%)

(a) 何謂 AVL Trees? (3%)

(b) 將元素加入 AVL Tree 時，若違反 AVL tree 的特性，則需進行旋轉(rotation)，請問在哪些狀況下需要旋轉? 又該如何旋轉? (請分項條列、舉例說明，否則不給分) (5%)

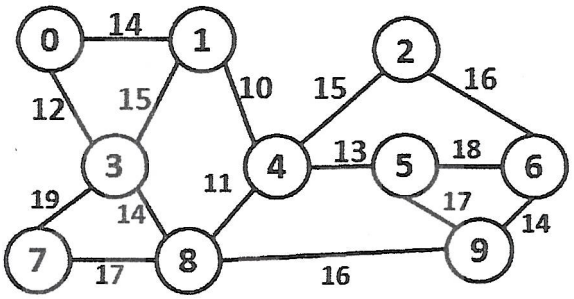
(c) 依序將以下鍵值加入一空的 AVL Tree，畫出此樹狀結構的改變過程。((b)答錯者，本小題不給分) (4%)

20, 19, 18, 17, 16, 15, 14, 7, 8, 9, 10

6. 參考以下無向圖(undirected graph)作答，節點(node)上的數字爲節點編號，邊(edge)上的數值爲成本。(13%)

[注意]: (1) 當有二個邊同時符合要求時，則以包含最小節點編號的邊爲優先，例如若同時考慮邊(2,6)與(8,9)，則以選取(2,6)爲優先。

(2) 需寫出過程，或註記"邊"被加入生成樹中的次序，否則不給分。



(a) 利用 Kruskal 演算法，找出此圖的最小成本生成樹，並計算其總成本。

(b) 利用 Prim 演算法，找出此圖的最小成本生成樹。

(c) 利用 Sollin 演算法，找出此圖的最小成本生成樹。

7. 參考以下以 Java 實作之 merge sort 程式碼回答問題: (15%)

(a) 請畫出一樹狀結構，以表示此程式遞迴呼叫的階層關係，並在各節點上標明被呼叫的次序。[提示]: 根節點的標籤爲 mergeSort(a,0,6)，次序爲 1。(5%)

(b) 倒數第二次與最後一次呼叫 merge()函數前，陣列 a[]的內容分別爲何? (6%)

(c) 以陣列 b[]爲例，配合圖形說明如何以非遞迴方式進行 merge sort。(4%)

```
public class TestMergeSort {
    public static void main(String[] args) {
        int b[] = {45, 28, 99, 30, 5, 54,15} ;
        mergeSort(b, 0, b.length-1) ;
    }
    public static void mergeSort(int[] a, int first, int last) {
        if (first < last) {
            int mid = (first+last)/2 ;
            mergeSort(a, first, mid) ;
            mergeSort(a, mid+1, last) ;
            merge(a, first, mid, last) ;
        }
    }
}
```

8. 參考以下 Java 程式回答問題: (20%)

(a) 請問程式一實作了哪一種排序法? 並分析其時間複雜度。(5%)

(b) 參考程式一，若將第 12 行刪除，並將第 9 行改爲 a[p+1]=x; break ;，則程式的輸出爲何? (5%)

(c) 程式二爲 quick sort 的實作，假設使用以下程式片段來呼叫，則第一次與第二次執行完第 11 行後，a[]陣列的內容分別爲何? (5%)

```
int[] a = {45, 28, 78, 5, 54, 15, 30} ;
quickSort(a, 0, a.length-1) ;
```

(d) 事實上，程式二中的 quickSort()函數存在著邏輯錯誤，請標出有錯的行號，並加以更正，再說明原因。(5%)

程式一

```
1  int a[] = {45, 28, 33, 99, 30} ;
2  int p ;
3  for (int i = 1; i<a.length; i++) {
4        int x = a[i];
5        for (p = i-1; p >= 0; p--) {
6              if (a[p]>x) {
7                    a[p+1]=a[p] ;
8              } else {
9                    break ;
10             }
11       }
12       a[p+1] = x ;
13 }
14 for (int i = 0 ; i<a.length; i++)
15       System.out.print(a[i]+" ") ;
```

程式二

```
1  public static void quickSort(int[] a, int m, int n) {
2      int p, i, j ;
3      if (m<n) {
4          i = m-1 ; j = n ;
5          p = a[n] ;
6          do{
7              do i++; while(a[i]<p) ;
8              do j--; while(a[j]>p) ;
9              if (i<j) swap(a, i, j) ; //將 a[i], a[j]互換
10         } while (i<j) ;
11         swap(a, n, i) ;
12         quickSort(a, m, i-1) ;
13         quickSort(a, i+1, n) ;
14     }
15 }
```