

類 科：資訊處理
科 目：資料結構
考試時間：2 小時

座號：_____

※注意：(一)禁止使用電子計算器。

(二)不必抄題，作答時請將試題題號及答案依照順序寫在試卷上，於本試題上作答者，不予計分。

(三)本科目除專門名詞或數理公式外，應使用本國文字作答。

一、以下是一中序運算式 (Infix expression) 轉換 (Convert) 成後序運算式 (Postfix expression) 的演算法

```
operstk = the empty stack;
while(not end of input){
    symb = next input character;
    if(symb is an operand)
        add symb to the postfix string;
    else{
        while(!empty(operstk) && precedence(stacktop(operstk), symb)){
            topsymb = pop(operstk);
            add topsymb to the postfix string;
        } /*end while*/
        if (empty(operstk) || symb != '(')
            push(operstk, symb);
        else
            topsymb = pop(operstk);
        } /*end else*/
    } /*end while*/
while(!empty(operstk)){
    topsymb = pop(operstk);
    add topsymb to the postfix string;
} /*end while*/
```

其中資料結構：

“operstk”：用來儲存運算子的堆疊 (Stack)；

“stacktop(operstk)”：表示 top 指標所指堆疊 operstk 的運算子；
程序 (Procedures) 或函數 (Functions)：

“empty(operstk)”：檢查堆疊 operstk 是否為空的布林函數；

“pop(operstk)”：從堆疊 operstk 中取出一運算子；

“push(operstk, symb)”：將運算子 symb 存入堆疊 operstk；

“precedence(op₁, op₂)”：布林函數，定義在一沒有左右括弧的中序運算式中，op₁ 運算子出現在 op₂ 運算子的左邊時，當 op₁ 運算子優先順序不低於 op₂ 運算子，則設定成 TRUE，否則為 FALSE。例如，我們給定 precedence('*', '+')=TRUE，precedence('+', '+')=TRUE，precedence('+', '*')=FALSE，為了處理運算式左右括弧，設定下列的 precedence：

```

precedence('(', op) = FALSE /*op 為任一運算子*/
precedence(op, '(') = FALSE /*op 為除')'外的任一運算子*/
precedence(op, ')') = TRUE  /*op 為除')'外的任一運算子*/
precedence(')', op) = undefined /*op 為任一運算子*/

```

 以中序運算式 $(2+3)*4$ 為例，執行上述演算法，依處理每一個運算子或運算元時，輸出 postfix string 及 operstk 內容為何(“eos”表示 end of string)? (25 分)

symbol	postfix string	operstk
(		
2		
+		
3		
)		
*		
4		
eos		

二、利用鏈結串列 (Linked list) 實做佇列 (Queues)，給予如下鏈結串列節點及佇列定義，front 指標指在串列第一個節點，rear 指標指在串列最後一個節點，請使用 C 語言完成 insert(pq, x) 程序，將整數值 x 加入 (Insert) 到佇列，程式需檢查佇列加入前是否為空的鏈結串列，可使用函數 getnode() 配置 (Allocate) 一新節點。(25 分)

```

struct node{
    int info;
    struct node *next;
};
typedef struct node *NODEPTR;

struct queue{
    NODEPTR front, rear;
};
struct queue q;

NODEPTR getnode()
{
    NODEPTR p;
    p = (NODEPTR)malloc(sizeof(struct node));
    return(p);
}

insert(pq, x)
struct queue *pq;
int x;
{
    NODEPTR p;
}

```

三、一個二元搜尋樹 (Binary search tree) 的前序追蹤 (Preorder traversal) 結果如下: 14, 4, 3, 9, 7, 5, 15, 18, 16, 17, 20

請建構此二元搜尋樹。接著利用如下 C 語言對二元樹節點的宣告, 使用 C 語言寫一遞迴程式 sortTree (NODEPTR tree), 輸入二元樹的根節點, 來處理此二元樹的節點資料, 並將資料依由小至大輸出。(25 分)

```
struct node{
    int info;
    struct node *left;
    struct node *right;
}
typedef struct node *NODEPTR;

void sortTree(NODEPTR tree){
}
```

四、用 $G = (V, E)$ 表示一個無方向性圖形, 其中 V 是點的集合, E 是一組節點 (Vertices) 形成邊及對應權重 (Weights) 所組成的集合。今有一圖形 $G = (V, E)$, $V = \{0, 1, 2, 3, 4, 5\}$, 圖形的邊與權重值以如下的定義儲存對應連接矩陣 (Adjacency matrix) 表示中的值

```
#define MAX_EDGES 100
typedef struct {
    int col;
    int row;
    int weight;
} edge;
edge a[MAX_EDGES];
```

已知陣列 a 儲存對應連接矩陣相連接邊的內容如下: $a = \{(3, 0, 2), (4, 0, 1), (5, 0, 20), (2, 1, 7), (5, 1, 24), (3, 2, 15), (4, 2, 10), (5, 2, 25), (4, 3, 3)\}$ 。請畫出陣列 a 所儲存的圖形, 然後, 利用 Prim 演算法從節點 0 開始依加入其它節點的順序, 畫出此圖之最小擴張樹 (Minimum spanning tree), 並計算其最低權重或成本值。(25 分)