

CH10

一、是非題

1. 陣列裡元素的資料型態可以不同。

【解答】 ×

2. 在程式執行時，陣列裡註標比較小的元素，會比註標大的元素更快拿到。

【解答】 ×

3. 環狀佇列是採用「先進先出」的順序。

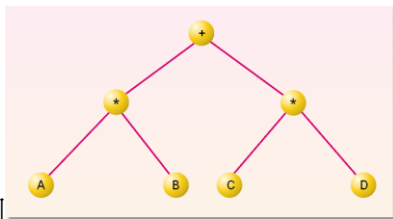
【解答】 ○

4. 環狀佇列裡宣告的每一個空間，都可以填入資料。

【解答】 ×

5. 樹只有一個根節點。

【解答】 ○



6. 在圖中的二元樹，其樹高為 3。

【解答】 (○)

二、選擇題

1. 以下何者代表 C 語言裡的空指標：

(A)null (B)nil (C)empty (D)not

【解答】 (A)

2. 以下何者的邏輯順序和實體順序不一定相同：

(A)鏈結串列 (B)一維陣列 (C)二維陣列 (D)以上皆是

【解答】 (A)

3. 以下哪種資料結構是採用「後進先出」的順序：

(A)陣列 (B)佇列 (C)堆疊 (D)環狀佇列

【解答】 (C)

4. 從根節點到樹中所有葉節點的最長可能路徑，稱作樹的

(A)高度 (B)階層 (C)根節點 (D)葉節點

【解答】 (A)

5. 在二元樹的探訪順序中，先探訪父節點、再探訪左子節點、最後探訪右子節點，稱作

(A)前序法 (B)中序法 (C)後序法 (D)循序法

【解答】 (A)

三、填充題

1. 假設系統在記憶體裡記錄多維陣列的方法，是先從第一列開始，然後接著記錄第二列，這種方式稱作_____。

【解答】 以列為主

2. 根據 C 語言的語法，若在宣告一個變數時前面加上_____符號，則該變數就是指標變數。

【解答】 *

3. 所謂的二元樹，就是每一個節點最多只有_____個子節點。

【解答】 2

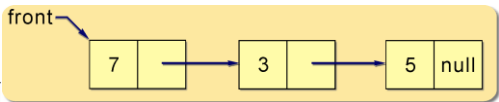
4. 將一個算數運算式以樹狀結構表示，此樹稱作_____。

【解答】 運算樹

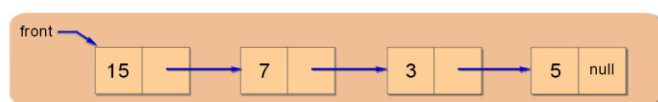
5. 在程序的本體中，又呼叫到自己本身，稱作_____程序。

【解答】 遞迴

四、問答題

1. 根據圖的鏈結串列，執行「insert(front, 15)」，畫出執行後的鏈結串列。

【詳解】



2. 利用第 7-3 節的堆疊宣告，改進程序"push"，要求在加入資料前，先判斷堆疊是否還有空位置。

【詳解】

```
void push (int data){
    if (top < 9)
    {
        top = top + 1;
        stack[top] = data;
    } else
    {
        printf("The stack is full.");
    }
}
```

3. 利用第 7-3 節的堆疊宣告，改進程序"pop"，要求在取出資料前，先判斷堆疊內是否有資料。

【詳解】

```
int pop( ){
    if (top >= 0)
    {
        top = top -1;
        return stack[top+1];
    }
    else
    {
        printf("The stack is empty.");
    }
}
```

4. 利用第 7-3 節的佇列宣告，改進程序"put"，要求在加入資料前，先判斷佇列是否還有空位置。

【詳解】

```
void put (int data){
    if (rear < 9)
    {
        rear = rear + 1;
        queue[rear] = data;
    } else
    {
        printf("The queue is full.");
    }
}
```

5. 利用第 7-3 節的佇列宣告，改進程序"get"，要求在取出資料前，先判斷佇列內是否有資料。

【詳解】

```
int get(){
    if (rear > front )
    {
        front = front +1;
        return queue[front];
    } else
    {
        printf("The queue is empty.");
    }
}
```

6. 討論何時使用陣列，何時使用鏈結串列。

【詳解】

如果資料不確定有多少，且時常動態增減，則較適宜使用鏈結串列。

7. 根據第 7-2 節 node 和 front 的定義，撰寫一個程序叫作 RemoveHead，把參數 front 指到的鏈結串列的第一個節點移除，然後回傳該節點所表示的資料（data）。

【詳解】

```
int RemoveHead(struct node *front)
{
    struct node *temp;
    temp = front;
    front = front->next;
    return(temp->data);
}
```

8. 討論何時使用堆疊，何時使用佇列。

【詳解】

如果我們希望先遇到的資料先處理，則使用佇列。反之，若希望先遇到的資料後處理，則使用堆疊。

9. 討論何時針對二元樹做後序法的探訪。

【詳解】

如果我們希望處理資料的順序，是先處理左子節點，接著是右節點，最後才處理父節點的話，則適用後序法。