1. (10%) What is the worst case time complexity of the followings.
   a) binary search on a sorted array with N integers
   b) search in a binary search tree with N nodes
   c) search in a sorted linked list with N nodes
   d) the bubble sort algorithm with N data elements
   e) searching in a MaxHeap with N data elements

   【Ans】:
   (a) O(\log_2 N)
   (b) O(N)
   (c) O(N)
   (d) O(N^2)
   (e) O(N)

2. (10%) Assume that the input data has N integers.
   a) What is the worst case time complexity of Insertion sort?
      What is the best case time complexity of Insertion sort?
      When will it happen?
   b) What is the worst case time complexity in data movement for Selection sort?
      What is the worst case time complexity in data comparison for Bubble sort?

   【Ans】:
   (a) worst case time complexity of Insertion sort: O(N^2)
       best casetime complexity of Insertion sort: O(N)
       when it happen: 當此 N 個 integers 已經是排序好的狀態
   (b) worst case time complexity in data movement for Selection sort: O(N)
       worst case time complexity in data comparison for Bubble sort: O(N^2)
3. (10%)
   a) Is Priority Queue a kind of Queue data structure? why?
   b) Please describe three kinds of implementations of Priority Queue and compare the time complexity in insertion and deletion of the different implementations.

【Ans】:
   (a) 不是，因為 Priority Queue 不會遵守 Queue Structure 先進先出的原則。
   (b) 3 種 Priority Queue 的 implementation:

<table>
<thead>
<tr>
<th>種類</th>
<th>Insertion</th>
<th>Deletion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Heap</td>
<td>(O(\log n))</td>
<td>(O(\log n))</td>
</tr>
<tr>
<td>Array</td>
<td>(O(1))</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Sorted Array</td>
<td>(O(n))</td>
<td>(O(1))</td>
</tr>
</tbody>
</table>

4. (10%)
   a) Please describe the algorithm to do Postfix expression evaluation.
   b) Please transform the following infix expression to prefix expression.

   \(( ( a + b) \times ( c - d ) - ( a - e ) ) \times ( c + d )\)

【Ans】:
   (a) Postfix expression evaluation:  
   假設算式 e 以’#‘作為結束
   procedure eval(e:expression);  
   begin  
   top := 0;  
   x := nexttoken(e);  
   while x <> ‘#’ do begin  
   if x is an operand then push(x)  
   else begin  
   pop(y1):  
   if x is a two-operand operator then pop (y2);  
   z := result of calculating x on y1(and y2);  
   push(z);  
   end;  
   x := nexttoken(e);  
   end;
5. (10%) a) Please write a C function to insert an integer into a linked list where the values in the data nodes are sorted. 
   b) Please write a C function to search in the linked list. 
   c) Please write a C function to delete a node from the linked list.

【Ans】:
(a) void insert ( node *head, int num)
{
    node * cptr = head;  //current ptr
    node * fptr = head;  //fptr point to previous node
    while( cptr != NULL)
    {
        if (num > cptr-> data )
        {
            cptr= fptr;
            break;
        }
        fptr = cptr;
        cptr = cptr -> next;
    }

    node *newnode = new node;
    if ( cptr == NULL)
        newnode -> next = head;
    else
    {
        if ( cptr-> next == NULL)
            cptr->next = newnode;
        else
        {
            newnode -> next = cptr->next;
            cptr -> next = newnode;
        }
    }
}
(b) node* search ( node* head , int num) {
    node *ptr = head;
    while ( ptr != NULL) {
        if (ptr -> data == num)
            return ptr;
        ptr = ptr->next;
    }
    return ptr;
}

(c) void delete ( node *head , int *dptr) // dptr point to delete node
{
    node *fptr; //fptr point to previous node
    if (dptr == head) //delete first node
        head = head -> next;
    else
    {
        fptr = head;
        while(fptr->next != dptr)
            fptr = fptr->next;
        if ( dptr-> next == NULL)  //delete last node
            fptr -> next = NULL;
        else
            fptr -> next = dptr->next;
    }
    free(dptr);
}
6. (10%)  
   a) What is a ‘tree’?  
   b) What is a ‘binary search tree’?  
   c) Given a binary tree, let $N_0$ denote number of leaf nodes, $N_1$ denote the number of degree one nodes, and $N_2$ denotes the number of degree 2 nodes, prove that $N_0 = N_2 + 1$.

   【Ans】:
   (a) 樹是由一個或多個節點構成的有限集合, 並且

      (1). 有一個特定的節點稱為樹根(root).

      (2). 其餘節點分成 n 個互斥集合, 這些互斥集合又都是一顆樹, 稱為樹根節點的子樹(subtree).

   (b) 在二元樹中任意節點 X, 左子樹中所有元素皆小於 X, 且右子樹中所有元素皆大於 X

      ex:

      \[
      \begin{array}{c}
      3 \\
      / \backslash \\
      2 \quad 5 \\
      / \quad / \\
      1 \quad 4 \quad 7
      \end{array}
      \]

   (c) 假設共有 n 點, $n = N_0 + N_1 + N_2$

      當 $n = 1$ 時, $N_0 = 1$, $N_1 = N_2 = 0$; $\Rightarrow N_0 = N_2 + 1$

      當 $n = 2$ 時, $N_0 = N_1 = 1$, $N_2 = 0$; $\Rightarrow N_0 = N_2 + 1$

      假設 $n = k$ 時, $N_0 = N_2 + 1$

      當 $n = K + 1$ 時, 若新點加在 degree = 1 的點下, 則 $N_1 = N_1 - 1$

      $N_0 = N_0 + 1$; $N_2 = N_2 + 1$; $\Rightarrow N_0 = N_2 + 1$

      若新點加在 degree = 0 的點下, 則 $N_1 = N_1 + 1$; $N_0, N_2$ 不變;
=> N₀ = N₂ + 1

則根據數學歸納法, N₀ = N₂ + 1 恆成立.

7. (10%) Please describe the algorithm to do Level Order tree traversal. What is the complexity of the algorithm.

【Ans】:
Level Order tree traversal: All nodes on level i are visited from leftmost to rightmost before the node of level i+1 is visited.

Void level_order (tree_pointer ptr)
{
    int front = rear = 0;
    tree_pointer queue[MAX_QUEUE_SIZE];
    if (!ptr)  // empty tree
        return;
    addq (front, &rear, ptr);
    for (; ; )
    {
        ptr = deleteq(&front, rear);
        if (ptr) {
            printf ("%d",ptr->data);
            if (ptr->left_child)
                addq(front, &rear, ptr->left_child);
            if (ptr->right_child)
                addq(front, &rear, ptr->right_child);
        }
        else
            break;
    }
}

The time complexity of this algorithm is O(N), where N is the number of the node.
8. (12%) Please describe an algorithm to solve the maze problem. Assume that the maze has \( N \times M \) entries, what is the worst case time complexity of the algorithm? Why?

\[ \text{Ans:} \]
initialize a stack to the maze’s entrance coordinates and direction to north;
while (stack is not empty)
{
    \(<\text{row}, \text{col}, \text{dir}> = \) delete from top of stack;
    while (there are more moves from current position)
    {
        \(<\text{next\_row}, \text{next\_col}> = \) coordinates of next move;
        dir = direction of move;
        if ((next\_row == EXIT\_ROW) \&\& (next\_col == EXIT\_COL))
            /* EXIT\_ROW and EXIT\_COL give the coordinates of the maze exit*/
            success;
        if (maze[next\_row][next\_col] == 0 \&\&
            mark[next\_row][next\_col] == 0)
        {
            mark[next\_row][next\_col] = 1;
            /* mark array is used to record the maze positions already checked */
            add <row, col, dir> to the top of the stack;
            // save current position and direction
            row = next\_row;
            col = next\_col;
            dir = north;
        }
    }
}
printf("No path found\n");

The worst case time complexity is \( O(NM) \), when each position within the maze is visited once.
9. **(10%)** Please describe an algorithm to do deletion on a binary search tree.

【Ans】:

想法:
先找到要刪除的節點,結果有 4 種 case:
CAES 1: 沒有左子樹且沒有右子樹
  a. 直接刪除節點

CASE 2: 只有左子樹時
  a. 將左子樹接到父節點

CASE 3: 只有右子樹時
  a. 將右子樹接到父節點

CASE 4: 同時有左右子樹時
  a. 把他的左子樹接到他的父節點
  b. 找出左子樹的最大節點
  c. 把原來的右子樹接到左子樹的最大節點

10. **(8%)** If we represent a m-way tree directly using direct child links, then how many links are void in a tree with N nodes? Please describe a tree representation scheme that will save the space for the links.

【Ans】:

N*m-(N-1)

化成二元樹即可節省 link 空間

Node structure:

<table>
<thead>
<tr>
<th>Lchild</th>
<th>Data</th>
<th>Rchild</th>
</tr>
</thead>
</table>

8
（1）長子繼承。兄弟左右：

5-way tree
A
|   |
A1-A2-A3-A4-A5
|   |
B1-B2-B3  C1-C2
|   |
D1-D2 E1

浪費了 14*5-(14-1)=57 個 link

（2）順時針旋轉 45 度：

A
/   
A1
/   /  \   
B1  A2  
/   /    /  
D1 B2 A3  
\  \    \    
D2 B3 A4
/  /      /  
E1 C1 A5
\  \  
C2

只浪費了 2*14-(14-1)=15 個 link