Tree.h

Post-order 部分參照範例提供的 in-order 函式，利用遞迴方式，以左子樹－右子樹－根節點的順序執行輸出。

Pre-order 部分同樣利用遞迴方式，以根節點－左子樹－右子樹的順序執行輸出。

Level-order 部分則參照講義，利用第六次作業的 Queue.h 完成輸出流程。

Insert 函式為建構二元搜尋樹的函式，將新元素依照二元搜尋樹左小右大的規則搜尋正確的位置並插入，利用一個 backup 變數儲存上一個節點，temp 儲存目前的節點，如果 temp 為 NULL 表示新元素不存在樹中且已經找到正確位置，依照新元素將與上一個節點的大小關係將元素插入左或右葉。若欲插入的元素已經存在，則直接返回。

Infix 函數的功用是將輸入的算式字串建成一個以運算元與運算子構成的二元樹。利用第六次作業的 postfix.h、token.h、stack.h 與先前提到的 Queue.h，稍加修改成可以套用至本次作業的形式。首先以 postfix 函式將輸入字串分解成個別的運算子與運算元並排列成後序算式。再判斷目前二元樹是否為空，若否則先將所有節點刪除並釋放空間並將根節點設為 NULL，是則開始建樹。因為後序運算式對應的為樹的 post-order，且運算元必為葉節點，故以一堆疊儲存韻算式，另一堆疊儲存節點位址，並由右而左建立二元樹。若目前節點的右子樹為空，則自堆疊談出一個元素放入，若此元素為運算子，則將目前節點位址存入堆疊後，將指向目前位址的指標移向右子節點，若為運算元則不儲存也不移動。若右子樹存在且左子樹為空，同樣自堆疊取出一元素放入，若元素為運算子，同樣將位址儲存並向左子節點移動，若為運算元，同樣不動作。若左右子樹皆存在，則將目前節點指標向上移動一層，方法為自位址堆疊彈出一位址給目前節點指標。重複至運算式堆疊為空為止。

Evaluate 函式為傳入二元樹的根節點，依此計算二元樹的數值，若根節點為運算元，則不存在任何子樹，因此直接傳回節點的數值。若根節點為運算子，則根據運算子以及遞迴算得的的左右子樹數值回傳相對應的運算結果，如根節點為加法運算子，回傳即為左子樹數值加上右子樹數值，減法運算子則回傳左子樹數值減去右子樹數值。

postfix.h、token.h、stack.h 與 Queue.h

主要修改部分為將 postfix 函式回傳值自 Queue 改為 Stack 以便利建樹

Main.cpp

開檔部分請依照檔案實際位址與執行時的目錄修改完整(含位址)檔名