資料結構 HW2

1. Insertion:

先檢查樹是否是空的。如果是空的則插入一node為root，將root與head和tail之間的thread連接好並結束。如果不是，則搜尋插入點的位置。新增一node。

如果插入點為左節點，則將parent node的left thread給新增的node的left，而原本parent node的left則接到新增的node，新增node的right則以thread方式接到parent node，如果parent node被head接上，因為新增node比parent node小，所以head改接上新增node。

如果插入點為右節點，則將parent node的right thread給新增的node的right，而原本parent node的right則接到新增的node，新增node的left則以thread方式接到parent node，如果parent node被tail接上，因為新增node比parent node大，所以tail改接上新增node。

1. Deletion:

先檢查樹上是否只剩root。如果是則刪除root節點並將所有變數設成初始化的值。如果不是則找出要刪除的node的位置。接下來以迴圈的方式重複找出該節點的predecessor或是successor，並將找到的點的number取代原本的節點的number，類似重複刪除節點的方式將要刪除的node傳到葉節點，因爲完全沒有改動中間樹的結構，所以完全不用維護任何thread或指標改動。接下來刪除node。

如果刪除點為左節點，則將刪除node的left thread給parent node的left，如果刪除node被head接上，則讓head改接parent node。

如果刪除點為右節點，則將刪除node的right thread給parent node的right，如果刪除node被tail接上，則讓tail改接parent node。

1. Inorder\_run:

先從head->right找出第一個要走的node。接下來以迴圈執行，當還沒走到tail時，輸出現在走到的node的值，再往該node的successor走。

如果該node的right是thread，node->right就是successor。

如果該node的right不是thread，則從node->right開始往左走到葉節點，該點就是successor。

1. Reverseorder\_run:

先從tail->left找出第一個要走的node。接下來以迴圈執行，當還沒走到head時，輸出現在走到的node的值，再往該node的predecessor走。

如果該node的left是thread，node->left就是predecessor。

如果該node的left不是thread，則從node->left開始往右走到葉節點，該點就是predecessor。