Data Structure
Homework 2 Report

學號：0416106　　姓名：彭敬樺

In this homework, we are to implement 4 function in the threaded binary search tree class. 4 function included is insertion, deletion, in-order run, and reverse in-order run.

In the insertion function is to create a new node in the tree that contains the number that wanted to be inserted. First, search for the appropriate parents by binary search tree rule that designated smaller number at the left node, larger number at the right node, and no data duplication inside the tree. If the data has already been in the tree before, we throw it away. Next, creating the new node to contains the number wanted to be inserted. If the tree is empty before insertion, the algorithm would be easy because we just have to point the root pointer to it and re-arrange head and tail pointer. If the tree is not empty originally, the insert it at the left node if it is smaller than its parent, or at the right node if it is larger than its parent.

In the deletion function, we first determine if the number that want to be deleted is in the tree or not. The searching is quite similar with what could be done in the insertion function. If it is not found, then the deletion process has failed. However, if it is found, the we could search the largest node at the left child or smallest node at the right child to take the node that we want to deleted and designate the node that take its place as the current node that wanted to be deleted. This loop process will occur again and again until if the current node that wanted to be deleted has no left and right child, then the process has finished. Now, we still need to actually delete the node, but we have to rearrange the left and right pointer of its parent to make sure that the parent node will not point to the deleted node(NULL) after deletion. Last thing but very important step is to rearrange the head and tail pointer to point the appropriate node.

In the in-order function, we need to traverse through all the node in the tree so that t will result in sequence of increasing number. This is more easily to be implemented because the threaded binary search tree has thread that will guide us easier through the entire tree. We start searching from the smallest number or the head pointer to the largest number or the tail pointer.

In the reverse order run, all we need to do is reverse the step that is implemented in the in-order function. We need to traverse from the tail pointer to the head pointer to have the effect of decreasing number result.

Below will be shown the result of 4 test case that is provided with the homework.

```
bsd1 [/u/cs/104/0416106/DS HW 2] -phua- ./a.out test1.txt
Change! Change myself into a cute mahou shoujo!!
The path: 2 3 5 6
Back! Back to the original life!!
The reverse path: 6 5 3 2
```

```
bsd1 [/u/cs/104/0416106/DS HW 2] -phua- ./a.out test2.txt
Change! Change myself into a cute mahou shoujo!!
The path: 1 2 3 4 5 6 7 8 9 10 11 12
Back! Back to the original life!!
The reverse path: 12
```

```
bsd1 [/u/cs/104/0416106/DS HW 2] -phua- ./a.out test3.txt
Change! Change myself into a cute mahou shoujo!!
The path: 1 2 3 4 5 6 7 8 9 10 11 12
Back! Back to the original life!!
The reverse path: 1
```

```
bsd1 [/u/cs/104/0416106/DS HW 2] -phua- ./a.out test4.txt
Change! Change myself into a cute mahou shoujo!!
The path: 14 16 23 24 29
Back! Back to the original life!!
The reverse path: 29 24 23 16 14
```