**Data Structure\_2071035 Lee Somin**

**Technical Report – sort\_binarysearch.cpp**

*Theorical Explanation of Functions in ‘sort\_binarysearch.cpp’*

**typedef struct TreeNode**

The structure contains the data of the node and the left, right nodes’ address.

**random**

Inputs: int n

Return: int (rand()\*rand())%n

This function is made for generating a randomly selected number from 0 to n by returning the remainder of (rand()\*rand()) divided by n. Since the range of function ‘rand()’ is not big enough, rand()\*rand() was put.

**insert\_node**

Inputs: TreeNode\*\* root, int key

Return: non

This function searches for the leaf node that is closest to the input key value. While loop compares the elements of tree with the key and goes to left or right node depending on the result of the comparison. It moves left when the key is smaller than the node, right when it isn’t. When reaching the leaf node, the while loop breaks and the new node is inserted as the child of the node found in the while loop. If ‘TreeNode\* p’ is NULL, which means that the tree is empty, it adds the new node as the root node of the tree.

**inorder\_print**

Inputs: TreeNode\* root

Return: non

This function prints the tree in inorder traversal by calling recursive call of itself when ‘TreeNode\* root’ is not NULL. The print order is left-root-right since it is inorder traversal.

**main**

In main function, memory is allocated for the input array and the output tree root pointer. Then, the root of the output tree is set to NULL. At the first for loop, input array elements are generated by 'random()'. Then, each of the input are put to the binary search tree at the second for loop. Finally, the sorted result are printed in ascending order by calling 'inorder\_print()' and printing out the binary search tree in inorder traversal.

(Result continued)

**Result:**

테이블이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명