**Data Structure\_2071035 Lee Somin**

**Technical Report – bst\_insertion\_deletion.cpp**

*Theorical Explanation of Functions in ‘bst\_insertion\_deletion.cpp’*

**typedef struct TreeNode**

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

**Delete\_node (case 3)**

Inputs: TreeNode \*\*root, int key

Return: non

At the beginning of the function, t is set as \*root, and p as NULL. Then, the t is updated to node with the target key, and p with the parent of t. Inside the else statement(case3), pred\_p is initialized as the node t, and pred is initialized as t-> left to find the predecessor at the left subtree of the target node. The while statement loops until there is no right child to pred and updates pred as its right child and pred\_p as current pred. After the while loop, pred is updated to the rightmost node of the left subtree. If the left child of the target node has no right subtree, it jumps to else statement and updates pred\_p’s left as the left child of the pred. If the left child of the target node has right subtree, the rightmost node is updated as its left child. Then, the key of the target node is changed to the key of the pred. As result, the target node’s value is changed to the value of predecessor and the value of the predecessor is updated to its left child.(there is no right child to predecessor because it’s the rightmost node of the left subtree of the target node.)

**Result:**

Key – 18

텍스트이(가) 표시된 사진

자동 생성된 설명

Key – 35

텍스트이(가) 표시된 사진

자동 생성된 설명

Key – 7

텍스트이(가) 표시된 사진

자동 생성된 설명