**Data Structure\_2071035 Lee Somin**

**Technical Report – Tree\_successor.cpp**

*Theorical Explanation of Functions in ‘Tree\_successor.cpp’*

**typedef struct TreeNode**

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

**tree\_successor**

Inputs: TreeNode\* p

Return: TreeNode\* p / TreeNode\* q

This function is made for finding the successor of the given tree node \*p. When the node is given, the function determines whether the right subtree of the given node is null. If it is not, the function updates p as the right node once and follows the left link of the tree until there is no node. Then, the function returns the p, which is the updated address of the leftmost node of the right subtree. When the right subtree of the given node is null, the function operates as next. If the parent node of the ‘p’ is null, the target node is the last node. So, there is no successor. If the node is left subtree of the parent node, the successor of the node is parent node. If the target node has no right subtree and is the right subtree of the parent node, it updates p as q and q as its parent node. When the loop ends, the function returns the successor (that has the target node as the left subtree) of the given node, ‘q’.

**main()**

input: non

return: non

In main function, the program assigns the parent nodes of the nodes in the tree. Then, it goes to the leftmost node. Then the program prints the data of the node and updates the next node(successor) of the current node to q using function ‘tree\_successor()’ until the successor of q is null. Using do-while statement, the loop is executed at least once.

**Result:**

텍스트이(가) 표시된 사진

자동 생성된 설명