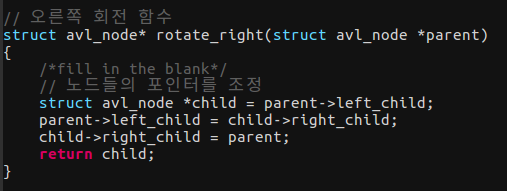
HW6: AVL TREE

2012003567

이상화

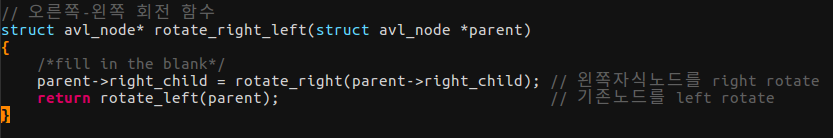
**1. 문제 해결의 주요 코드**

(1) struct avl\_node\* rotate\_right(struct avl\_node \*parent)



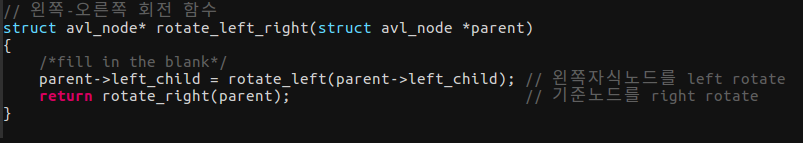
parent를 기준으로 right rotate시키는 함수이다. left child를 parent로 만들기 위해 left child의 right child를 parent의 left child로 붙히고 left child의 right child로 parent를 달아준뒤 left child를 반환한다.

(2) struct avl\_node\* rotate\_right\_left(struct avl\_node \*parent)



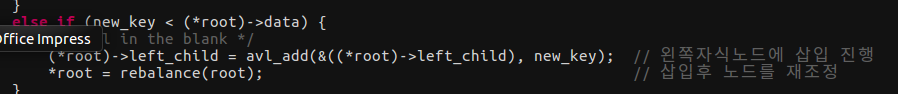
parent의 right child를 기준으로 right rotate 시킨 뒤에 parent를 다시 left rotate 시켜준뒤 이를 반환한다.

(3) struct avl\_node\* rotate\_left\_right(struct avl\_node \*parent)



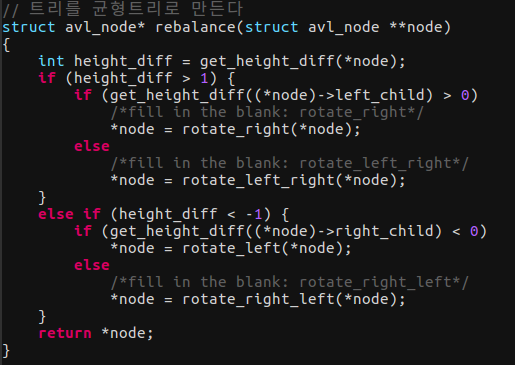
parent의 left child를 기준으로 left rotate 시킨 뒤에 parent를 다시 right rotate 시켜준뒤 이를 반환한다.

(4) struct avl\_node \* avl\_add(struct avl\_node \*\*root, int new\_key)



삽입시 삽입하려는 key보다 큰 key를 가진 node를 만나면 node의 left child로 삽입을 진행해야 한다, 삽입 뒤에 rebalance 함수를 통해 밸런스를 맞춰준다.

(5) struct avl\_node\* rebalance(struct avl\_node \*\*node)



node의 balance가 깨졌을 시에 rebalance 해주는 함수이다. height\_diff는 left subtree의 height-right subtree의 height이므로 height\_diff>1인 경우는 left subtree에 삽입이 이루어져서 balance가 깨진 경우이다. 이 때 left child의 height\_diff가 양수인 경우엔 right rotate만 해주면 되고 음수인 경우엔 left child를 기준으로 left rotate를 해주고 다시 right rotate를 해주면 된다. height\_diff < -1인 경우엔 이와 정확히 반대로 해주면 된다.

**2. 실행결과**

