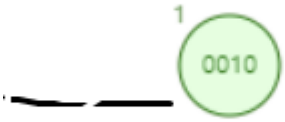
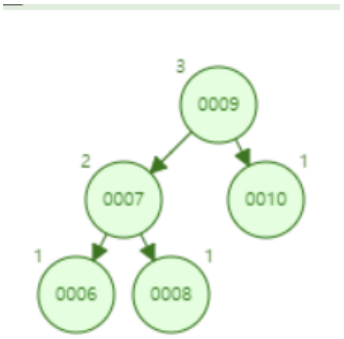
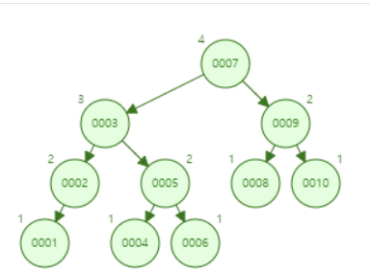
InsertTest 보고서

InsertTest 0

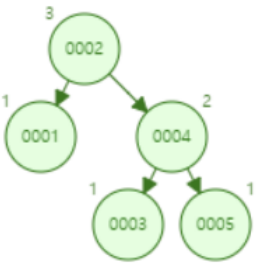


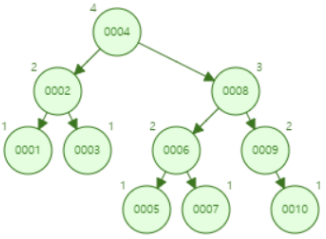
InsertTest 1



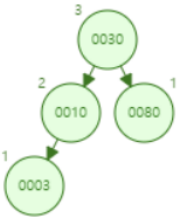


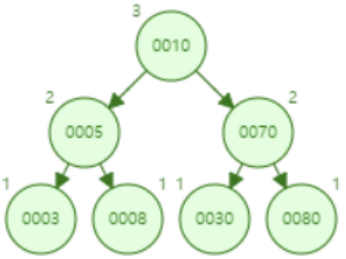
InsertTest 2



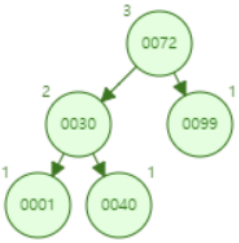


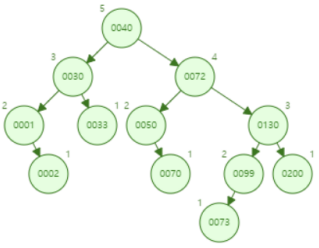
InsertTest 3

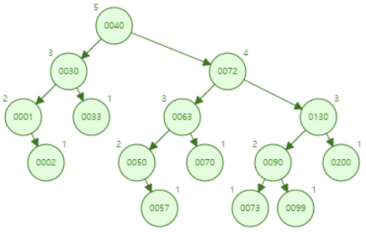


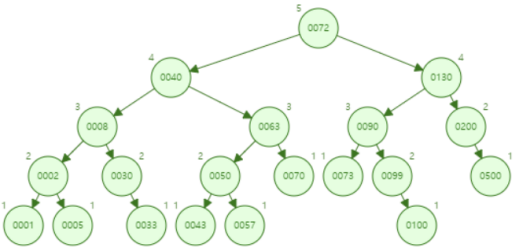


InsertTest 4









테스트 코드 커버리지 결과

```
      -:    1:#include "avl_tree.h"
      -:    2:#include "tree_node.h"
      -:    3:#include<algorithm>
      -:    4:using namespace std;
      -:    5:
    128:    6:int AVLTree::getBalance(TreeNode* node) { // Balance Factor(BF) 계산하는 함수
    128:    7:    if (node == nullptr)
  #####:    8:        return 0;
    128:    9:    int leftNodeBalancedFactor = (node->leftNode() == nullptr) ? -1 : node->leftNode()->height();
    128:   10:    int rightNodeBalancedFactor = (node->rightNode() == nullptr) ? -1 : node->rightNode()->height();
    128:   11:    return leftNodeBalancedFactor - rightNodeBalancedFactor; // 좌우 자식 깊이를 비교해 BF 리턴
      -:   12:}
      -:   13:
     14:   14:TreeNode* AVLTree:: RotateRight(TreeNode* z) { // y는 z의 왼쪽 자식 노드, x는 y의 왼쪽 자식 노드로 설정, z를 중심으로 오른쪽 회전
     14:   15:    TreeNode *y = z->leftNode();
     14:   16:    TreeNode *T2 = y->rightNode(); // T2는 y의 오른쪽 자식
      -:   17:
      -:   18:    // right 회전 수행
     14:   19:    y->setRightNode(z);// y 노드의 오른쪽 자식 노드를 z노드로 변경
     14:   20:    z->setLeftNode(T2); // z 노드의 왼쪽 자식 노드를 y노드 오른쪽 서브트리(T2)로 변경
      -:   21:    // 위치가 바뀌었으므로 노드 높이 갱신
      -:   22:
      -:   23:    //사이즈 변경
     14:   24:    z->setSize(1 + SubtreeSize(z->leftNode()) + SubtreeSize(z->rightNode()));
     14:   25:    y->setSize(1 + SubtreeSize(y->leftNode()) + SubtreeSize(y->rightNode()));
      -:   26:
      -:   27:
     14:   28:    int zLeftHeight = (z->leftNode() != nullptr) ? z->leftNode()->height() : -1;
     14:   29:    int zRightHeight = (z->rightNode() != nullptr) ? z->rightNode()->height() : -1;
     14:   30:    z->setHeight(1 + max(zLeftHeight, zRightHeight));
      -:   31:
     14:   32:    int yLeftHeight = (y->leftNode() != nullptr) ? y->leftNode()->height() : -1;
    14*:   33:    int yRightHeight = (y->rightNode() != nullptr) ? y->rightNode()->height() : -1;
     14:   34:    y->setHeight(1 + max(yLeftHeight, yRightHeight));
      -:   35:
      -:   36:    // 새로운 루트 노드 y를 반환
     14:   37:    return y;
      -:   38:}
      -:   39:
```

```
 17:   40:TreeNode* AVLTree:: RotateLeft(TreeNode* z) { // y는 z의 오른쪽 자식 노드이고, x는 오른쪽 자식 노드인 경우 z를 중심으로 왼쪽 회전
 17:   41:    TreeNode *y = z->rightNode();
 17:   42:    TreeNode *T2 = y->leftNode();
  -:   43:
  -:   44:    // left 회전 수행
 17:   45:    y->setLeftNode(z); // y노드의 왼쪽 자식 노드를 z노드로 변경
 17:   46:    z->setRightNode(T2); // z노드의 오른쪽 자식 노드를 y노드 왼쪽 서브트리(T2)로 변경
  -:   47:
  -:   48:    //사이즈 변경
 17:   49:    z->setSize(1 + SubtreeSize(z->leftNode()) + SubtreeSize(z->rightNode()));
 17:   50:    y->setSize(1 + SubtreeSize(y->leftNode()) + SubtreeSize(y->rightNode()));
  -:   51:
  -:   52:
  -:   53:    // 위치가 바뀌었으므로 노드 높이 갱신
 17:   54:    int zLeftHeight = (z->leftNode() != nullptr) ? z->leftNode()->height() : -1;
 17:   55:    int zRightHeight = (z->rightNode() != nullptr) ? z->rightNode()->height() : -1;
 17:   56:    z->setHeight(1 + max(zLeftHeight, zRightHeight));
  -:   57:
17*:   58:    int yLeftHeight = (y->leftNode() != nullptr) ? y->leftNode()->height() : -1;
 17:   59:    int yRightHeight = (y->rightNode() != nullptr) ? y->rightNode()->height() : -1;
 17:   60:    y->setHeight(1 + max(yLeftHeight, yRightHeight));
  -:   61:
  -:   62:    // 새로운 루트 노드 y를 반환
 17:   63:    return y;
  -:   64:}
  -:   65:
 48:   66:int AVLTree::Insert(int key) {
  -:   67:
 48:   68:    root_ = InsertRecursive(root_, key); // 루트에서부터 시작해 재귀적으로 키 값 삽입
 48:   69:    return Find(key);
  -:   70:
  -:   71:}
176:   72:TreeNode* AVLTree:: InsertRecursive(TreeNode* node, int key) {
176:   73:    if (node == nullptr) { // 노드가 널포인터 값일 경우 전체 노드의 개수를 1 증가시키고, TreeNode 생성
 48:   74:        ++total_node_cnt_;
 48:   75:        return new TreeNode(key, nullptr, nullptr, nullptr);
  -:   76:    }
128:   77:    if (key < node->key()) {
 61:   78:        node->setLeftNode(InsertRecursive(node->leftNode(), key));
 67:   79:    } else if (key > node->key()) {
 67:   80:        node->setRightNode(InsertRecursive(node->rightNode(), key));
```

```
    67:   80:        node->setRightNode(InsertRecursive(node->rightNode(), key));
     -:   81:    } else {
 #####:   82:        return node;
     -:   83:    }
     -:   84:
   128:   85:    int leftHeight = (node->leftNode() != nullptr) ? node->leftNode()->height() : -1;
   128:   86:    int rightHeight = (node->rightNode() != nullptr) ? node->rightNode()->height() : -1;
     -:   87:
   128:   88:    node->setHeight(1 + max(leftHeight, rightHeight));
     -:   89:
   128:   90:    int leftSize = (node->leftNode() != nullptr) ? node->leftNode()->size() : 0;
   128:   91:    int rightSize = (node->rightNode() != nullptr) ? node->rightNode()->size() : 0;
   128:   92:    node->setSize(1 + leftSize + rightSize);
     -:   93:
   128:   94:    return Balancing(node, key);
     -:   95:}
     -:   96:
   128:   97:TreeNode* AVLTree::Balancing(TreeNode* node, int key) { // BF를 이용해 회전로직을 구현
   128:   98:    int balance = getBalance(node); // 노드 밸런스 유지
     -:   99:
     -:  100:    // LL (Left Left, right rotation 수행하여 균형을 맞춤)
   128:  101:    if (balance > 1 && key < node->leftNode()->key()) {
     6:  102:        node = RotateRight(node);
     -:  103:    }
     -:  104:
     -:  105:    // RR (Right Right, left rotation 수행하여 균형을 맞춤)
   122:  106:    else if (balance < -1 && key > node->rightNode()->key())
     9:  107:        node = RotateLeft(node);
     -:  108:
     -:  109:    // LR (Left Right 순으로 총 두번의 rotation 수행하여 균형을 맞춤)
   113:  110:    else if (balance > 1 && key > node->leftNode()->key()) {
     5:  111:        node->setLeftNode(RotateLeft(node->leftNode()));
     5:  112:        node =  RotateRight(node);
     -:  113:    }
     -:  114:    // RL (Right, Left 순으로 총 두번의 rotation 수행하여 균형을 맞춤)
   108:  115:    else if (balance < -1 && key < node->rightNode()->key()) {
     3:  116:        node->setRightNode(RotateRight(node->rightNode()));
     3:  117:        node= RotateLeft(node);
     -:  118:    }
   128:  119:    return node;
```

테스트 수행 결과

```
C:\Users\edgar\CLionProjects\INHA_OSAP_003_babamba\cmake-build-debug\insertTest.exe
[==========] Running 5 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 5 tests from SET_AVLTEST
[ RUN      ] SET_AVLTEST.InsertTest0
[       OK ] SET_AVLTEST.InsertTest0 (0 ms)
[ RUN      ] SET_AVLTEST.InsertTest1
[       OK ] SET_AVLTEST.InsertTest1 (0 ms)
[ RUN      ] SET_AVLTEST.InsertTest2
[       OK ] SET_AVLTEST.InsertTest2 (0 ms)
[ RUN      ] SET_AVLTEST.InsertTest3
[       OK ] SET_AVLTEST.InsertTest3 (0 ms)
[ RUN      ] SET_AVLTEST.InsertTest4
[       OK ] SET_AVLTEST.InsertTest4 (0 ms)
[----------] 5 tests from SET_AVLTEST (20 ms total)

[----------] Global test environment tear-down
[==========] 5 tests from 1 test suite ran. (36 ms total)
[  PASSED  ] 5 tests.

Process finished with exit code 0
```