AVL tree

- **AVL의 삽입 알고리즘 insertAVL(T, newKey)을 구현하시오.**
  - 단 알고리즘은 다음 함수를 이용해 구현하시오.
  - insertBST(T, newKey)
  - checkBalance(T, newKey, rotationType, p, q)
    - newKey의 위치를 검색.
    - newKey부터 루트까지 BF를 다시 계산함.
    - 불균형이 발생하면, 발생한 노드 p와 그 노드의 부모 노드 q, 그리고 회전의 종류 rotationType을 리턴.
    - 불균형이 발생하지 않으면, rotationType은 “NO”로 하고, p와 q는 널 값을 리턴.
  - rotateTree(T, rotationType, p, q)
    - 회전의 종류 rotationType에 따라, p를 루트로 하는 서브트리를 회전함.

- **출력을 위해 트리의 inorder 순회 알고리즘을 구현하시오**
  - inorderBST(T)를 그대로 이용.

- **노드 삽입 순서**
  - 40 11 77 33 20 90 99 70 88 80 66 10 22 30 44 55 50 60 100 28 18 9 5 17 6 3 1 4 2 7 8 10 12 13 14 16 15

- **출력**
  - 삽입과 삭제 모두 노드 하나를 삽입 삭제할 때마다, 회전의 종류(LL/LR/RL/RR) 그리고 트리의 inorder 순회 순서를 출력하시오.
  - 따라서 다음과 같은 main 루틴을 이용하시오.

- **Main**
  - T ← null;

  - // 삽입
  - insertAVL(T, 40); print(rotationType); inorderBST(T);
  - insertAVL(T, 11); print(rotationType); inorderBST(T);
  - . . . .
  - insertAVL(T, 100); print(rotationType); inorderBST(T);

- **제출물**
  - 표지
  - 출력 결과 (화면 덤프 혹은 출력 파일 프린트)
  - 프로그램 소스