Binary search tree (BST)

- **BST의 삽입 알고리즘 insertBST(T, newKey)을 구현하시오.**
  - 단 알고리즘은 수업 슬라이드에서 설명한 다음 함수를 이용해 구현하시오.
  - getNode()

- **BST의 삭제 알고리즘 deleteBST(T, deleteKey)을 구현하시오.**
  - 단 알고리즘은 수업 슬라이드에서 설명한 다음 함수를 이용해 구현하시오.
  - height(T)
  - noNodes(T)
  - maxNode(T)
  - minNode(T)

- **출력을 위해 트리의 inorder 순회 알고리즘을 구현하시오**
  - inorderBST(T)

- **노드 삽입 순서**
  - 40 11 77 33 20 90 99 70 88 80 66 10 22 30 44 55 50 60 100
- **노드 삭제 순서**
  - 1. 삽입 순서와 동일
  - 2. 삽입 순서와 역순
- **출력**
  - 삽입과 삭제 모두 노드 하나를 삽입 삭제할 때마다 트리의 inorder 순회 순서를 출력하시오.
  - 따라서 다음과 같은 main 루틴을 이용하시오.

- **Main**
  - T ← null;
  - // 삽입
  - insertBST(T, 40); inorderBST(T);
  - insertBST(T, 11); inorderBST(T);
  - . . . .
  - insertBST(T, 100); inorderBST(T);
  - // 삭제 1
  - deleteBST(T, 40); inorderBST(T);
  - deleteBST(T, 11); inorderBST(T);
  - . . . .
  - deleteBST(T, 100); inorderBST(T);

- T ← null;

- // 재삽입
- insertBST(T, 40); inorderBST(T);
- insertBST(T, 11); inorderBST(T);
- . . . .
- insertBST(T, 100); inorderBST(T);

- // 삭제 2
- deleteBST(T, 100); inorderBST(T);
- deleteBST(T, 60); inorderBST(T);
- . . . .
- deleteBST(T, 40); inorderBST(T);

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