Homework6: BST(Binary Search Tree) vs. BBST(Balanced Binary Search Tree)
Data Structures, Konkuk University

수업 시간에 배운, 일반 binary search tree와 AVL tree (Balanced Binary Search Tree)를 구현하시오. 즉, 각각의 data structure를 위한 **insert, find** 함수를 구현하시오. find의 경우는 두 트리가 같은 구현을 이용할 수 있습니다.

1. 위의 두 tree에 대해서, 1~10만까지의 integer를 random하게 발생시켜서 insert하는 작업을 10만번 수행한 후 수행 시간을 측정하여 1회 평균 수행시간을 구하시오. 그리고, 1~10만까지의 숫자를 random하게 발생시켜 find하는 작업을 10만번 수행하는 실험을 하여 수행시간을 측정하여, 1회 평균 수행시간을 구하시오. 이러한 실험을 20만, 30만, ..., 100만까지 수행(범위도 같이 증가)하여, 1회 수행 시간에 대한 time complexity의 간단한 결론을 제시하시오.

2. 위의 두 tree에 대해서, 1, 2, 3, ..., 1000까지의 integer를 순서대로 insert하는 작업을 수행한 후, 수행 시간을 측정하여, 1회 평균 수행시간을 구하시오. 그리고, 1~1000사이의 숫자를 random하게 발생시켜 find하는 작업을 1000번 수행하여 수행 시간을 측정하여, 1회 평균 수행시간을 구하시오. 이러한 실험을 2000번, 3000번, 4000회, 5000회(범위도 같이 증가)를 실하여, 1회 수행시간에 대한 time complexity의 대한 간단한 결론을 제시하시오. Tree안의 노드 수가 많아지면, recursive call의 depth가 길어지고, 이에 따라 stack overflow 현상이 발생합니다. 4000정도에서 stack overflow가 일어나는 경우는 숫자 범위와 실행 수를 줄여서 실행하여도 됩니다. (예: 4000까지만 실험 수행). Visual Studio를 이용하는 경우, Debug 모드가 아니라 Release 모드로 빌드를 하면 stack overflow 문제를 어느 정도 해결할 수 있습니다.

제출방법:

(1) 소스 파일(***.c 또는 ***.cpp)과 보고서를 ecampus에 제출하시오. 소스 파일 맨 윗쪽에 코멘트로 학번과 이름, 실행환경(Windows, Mac, Linux)을 명시할 것. **작성한 function들이 몇 번 문제에 해당하는 function인지 대한 설명을 코멘트를 반드시 남기고, 수행시간을 printf할 때도 어떤 문제의 수행시간인지를 구별할 수 있도록 하시오.**