CSci 115 Spring 2016
Lab 5
(due Tue Mar 8, 6:00pm)

Devise a variation of the BST data structure that is specialized to `int`'s (that is, no templates) and that supports the following operations (where *n* is the number of elements in the tree):

- a constructor that creates an empty tree in constant time;
- insert a key into the tree in O(log n) time (assuming that the tree stays balanced);
- return the kth smallest key in the tree in O(log n) time (note that this running time does not depend on k);
- print all the keys, in order, between (and including) two given keys, visiting as few nodes of the tree as possible; this function should return the number of nodes visited during the operation (where “visited” means you looked at the value in that node).

Create a header file that exports only what is needed to use this data structure, and an implementation of the data structure in a separate `cpp` file. Write a driver program that includes the header file and runs a comprehensive series of tests on the implementation, indicating the status of each test. The driver and the implementation should be compilable separately so that they can be linked together to produce an executable.

Note that, in order to be able to return the kth smallest key in the tree (for any given value of k) in log(n) time, you will have to store additional information into the nodes of the tree. Make sure you state, in comments, the invariant that this extra information satisfies, and, in comments inside the insert function, some explanation for why we should believe that your insert function maintains this invariant.

To turn in: `bst.h`, `bst.cpp`, `driver.cpp`