This lab is about BINARY SEARCH TREE, so the nodes have the ordering property.

These must be completed and shown to your lab TA either by the end of this lab, or by the start of your next lab.

Download the binary search tree code from the course web page under Lab 5. There are a number of failing tests which you must pass. You can run the tests with:

```
> ./bst
```

You also have the ability to write your own testing/debugging code and run that instead. To do that, just supply your test keys as command line arguments:

```
> ./bst 5 3 2 1 6 8 4 7 9
```

1. To fix the failing tests, first implement the following functions in `bst.cpp`:

```
/**
 * Returns the height of node x.
 */
int height( Node* x );

/**
 * Traverse a tree rooted at rootNode in-order and use 'v' to visit each node.
 */
void in_order( Node*& rootNode, int level, Visitor& v );

/**
 * Traverse a tree rooted at rootNode pre-order and use 'v' to visit each node.
 */
void pre_order( Node*& rootNode, int level, Visitor& v );

/**
 * Traverse a tree rooted at rootNode post-order and use 'v' to visit each node.
 */
void post_order( Node*& rootNode, int level, Visitor& v );
```

2. When that is done, you can complete the missing portion(case 2) of `delete_node` in `bst.cpp`:

```
/**
 * Deletes a node containing 'key' in the tree rooted at 'root'.
 */
bool delete_node(Node*& root, KType key);
```

3. Which functions would change if the Nodes were part of a binary tree that didn't have the search tree property (the invariant that requires `left < parent < right`)?

4. Complete the `find_second_smallest(Node* rootNode)` function to return the node with the second smallest key.