Binary Search Trees

These must be completed and shown to your lab TA either by the end of the lab session, or at the start of your next lab. You may work in groups of up to two people.

Download the binary search tree code from the course webpage under Lab 5. You can compile them using `make` in Linux, or you may add them to a project in an IDE of your choice. The default test program has over 40 failing tests which you must pass. You can also write your own testing/debugging code and run that instead by supplying your test keys as command line arguments, e.g. `./bst 5 3 2 1 6 8 4 7 9`

Your own test code can be added to `runMain()`. To fix the failing tests, implement the marked functions in `bst.cpp` (many of them can be implemented recursively – don't forget your base case(s)):

- `numNodes`
- `numLeaves`
- `height`
  - a tree with 0 or 1 nodes has height 0
- `depth`
- `in_order`
- `pre_order`
- `post_order`
- `delete_node`
  - use the predecessor for the 2-child removal case
  - pay attention to different cases where the predecessor is a left child or a right child

Errors in one function may cause other functions to misbehave! Use the "Step into" feature of your debugger to trace your program execution if you encounter unexpected behaviour.

Please refer to last week's lesson on binary search trees for illustrations of the BST terminology and operations.

Show your work to your TA before you leave, or at the start of the next lab, or you will not receive credit for the lab!