Building and usage:

To build just type make.
The serial executable is “serial” and the parallel is “parallel”.

Balancing the tree:

To check if the tree is balanced I find the depth of the left subtree and the right subtree starting at the root node. If the difference between these two is greater than 1 then the tree is said to be unbalanced.

To balance the tree I convert it to a sorted list and then create a new tree by inserting elements of the list starting at the middle and moving out from there. For example say the list has ten elements. First element 5 is inserted, then element 4, then element 6 and so on. This ensures that the tree is balanced once all elements have been inserted.

Deleting nodes:

When deleting nodes care had to be taken to ensure that all child/parent links are preserved. In addition if the nodes has left and right subtrees it has to be treated as a special case. In this case the node is not actually deleted, instead the node with the minimum value in its subtree tree is found and this is deleted. The minimum value is then inserted into the original node to be deleted. This approach is described in the following link:
http://www.algolist.net/Data_structures/Binary_search_tree/Removal

Using pthreads:

Not much to be said here. Each thread sleeps for a time then tries to acquire a mutex before doing its work. I have set the sleep times so that the insertion thread runs more often than the deletion thread, which in turn runs more often than the balancing thread.

Random numbers are inserted into the tree. Because of the large range of values rand() provides it is assumed that no numbers will be identical. To delete numbers I wrote some code to get a random node from the tree which is then deleted.