Implementation of Tree

This is the implementation of balance tree, which can store integers on the leaves. In order to make it be the balanced tree, the difference of heights of left and right sub-trees cannot be more than one, and both of left and right sub-trees should be balanced. We construct a new data constructor named Tree with two parts: Leaf and Node. For function balanceTree, we do some edge condition analysis, when the list length is equal to one, then there is only one leaf; when the list length is equal to two, the tree should be one node and two leaf. We create a split function which can split a list to tuple of list. The first list in the tuple should be the left sub-tree, and the second list in tuple should the right sub-tree. Finally, we do the recursion on left sub-tree and right sub-tree.

Moreover, we also implement inserting, removing, searching a leaf and find the minimum and maximum valves on the true from a tree. We also write a print function which can print tree structure.

Trouble we faced:

When we face some syntax trouble, which always confused us. The process of doing project help us to be familiar with Haskell.

Reference:

The tree data structure:

<https://www.codementor.io/haskell/tutorial/monoids-fingertrees-implement-abstract-data>

<http://zvon.org/other/haskell/Outputprelude/replicate_f.html>

<http://stackoverflow.com/questions/15080008/how-does-one-pretty-print-recursion-depth-in-haskell>