Exercise (Friday) — Experimenting with QuickSpec

For these exercises, you will need to install QuickSpec. Do so using

```
cabal update
```

(update the list of libraries that cabal knows about), followed by

```
cabal install quickspec --lib
```

This may take a little while to run, depending on how many of the necessary libraries you have already installed.

QuickSpec depends on QuickCheck, and you need to have the same version of QuickCheck installed as QuickSpec is using. If you encounter unexpected compile-time errors, such as ’No instance Arbitrary Tree’, then this may be a sign that you are using two versions of QuickCheck at the same time.

1. You are provided with the files **`Int.hs`**, **`Rational.hs`**, and **`Float.hs`**, containing the code discussed in the lecture. Add multiplication (**`*`**) and **`1`** to the signatures and run QuickSpec, to see if you get the expected laws. When you run QuickSpec on the joint signature, do you get an informative specification of floating point multiplication?

2. You are also provided with **`BSTSpec.hs`**, containing the signatures for binary search trees discussed in the lecture, and (another copy of Tuesday’s) files defining binary search trees (**`BST.hs`**) along with eight buggy versions.

    a. The signatures provided only include **`find`** and **`insert`**. Extend them by adding **`delete`**, and see how the results from QuickSpec change. (In particular, compare the laws involving **`delete`** that you get using equality and tree equivalence).

    b. **`BST4`** and **`BST5`** define buggy versions of delete. Can you characterise the bugs just from the QuickSpec output? (Compare the laws generated for the correct version with those generated for the buggy versions).