Building Trees

(2 points.) To start, run `python new-homework.py [language] tree-builder`

Background. In this assignment, we'll build a data structure similar to dictionaries or associative arrays: it stores many items, each of which has a key (a label) and a value. Once items are stored in the data structure, we can quickly look them up by using their key to fetch the associated value.

Normally, dictionaries are implemented using hash tables, as we discussed in class. But in some cases dictionaries are implemented using other strategies. One possible strategy is to use a binary tree.

For us to build a dictionary using a binary tree, the keys must be *ordered*: it should be possible and meaningful to ask if `key1 < key2` for any two keys in the dictionary. Using a binary tree lets us look up items by key, but it also lets us find all items with keys in a certain range, or easily produce all items in order, sorted by their keys. Sometimes this is quite useful.

We'll call a dictionary implemented this way a *treemap*.

Task. There are two sets of files in the `Resources/` directory for this assignment. The files `tree_builder.R` and `tree_builder.py` are the files you will work on (pick R or Python, whichever you prefer), and are mostly filled out—but are missing key details, marked with comments labeled `TODO: Fill in`. Fill in these details and make the code work. A few unit tests are provided so you understand what the code should do, but ***you should write more***.

The files `tree_builder_util.R` and `tree_builder_util.py` contain functions you'll be using. You should *not* need to modify anything in these files, nor should you have to read the code they contain in any detail, only the documentation comments as needed. Don't worry about how the code in here works.

You should proceed in order:

1. Copy the R or Python files, whichever you choose, from the `Resources/` directory into the main `tree-builder/` directory for this assignment. Do this first.

2. Fill in `treemap_insert`, ensure it runs on some examples, and print out the resulting treemap to make sure it looks reasonable.

3. Now fill in `treemap_search`. Once you've filled it in, run the unit tests and ensure they pass. (The tests for `treemap_search_between` will fail, but that's fine.) Write any additional tests you need to be confident in your answer.

4. Now fill in `treemap_search_between`. Once you've filled it in, again run the unit tests and ensure they pass. Write any additional tests you need to be confident in your answer.

Questions. Briefly answer the following questions in comments on your GitHub pull request.

1. In $O(n)$ notation, about how long does it take to find a specific key in the tree when there are n items stored?

Requirements.

- ☐ Fill in all the `TODO` items in the R or Python code, whichever you prefer.
- ☐ Answer the questions above in your pull request.
- ☐ Supply a test suite for your functions, beyond the ones provided for you. *Ensure all tests pass.*