Lab 10: Binary trees

Create a separate file for each question. Keep them in your "Labs" folder, with the name `liqj` for Lab *i*, Question *j*.

Download the headers for each function from the file `labinterface10.rkt` linked off the "Labs" page on the course Web site.

After you have completed a question (except class exercises), including creating tests for it, you can obtain feedback by submitting it and requesting a public test. Follow the instructions given in the Style Guide.

For your convenience, the structure definitions and the data from the example used in lecture can be found in the teachpack `taxon.rkt` linked off the "Labs" page. Instructions for loading the taxon teachpack (and other teachpacks) are available in the Helpful tips section "Loading a teachpack".

The tests will use the teachpack. Do not paste material from the teachpack into your lab files, or auto-testing will fail (constants will be defined twice).

Language level: Beginning Student with List Abbreviations

1. *[Class exercise with lab instructor assistance]* Create a function *count-funs* that consumes a binary arithmetic expression and produces the number of functions in that expression.

2. Create a function *is-ancestor-of?* that consumes a taxon and a string and produces true if the taxon is the ancestor of a taxon with the given name. A taxon is its own ancestor.

3. Create a function *desc-list* that consumes a taxon and produces a list of strings consisting of the names of the modern descendants of the taxon. A taxon is considered to be its own descendant.

4. Create a function *odd-lengths* that consumes a binary search tree and produces a copy of the tree in which each odd-length value is replaced by the string `"odd"`.

5. Create a function *leftmost* that consumes a non-empty binary search tree and produces the leftmost node in the tree.

6. *Optional open-ended questions*

(a) For a binary search tree to be a useful way of storing data, it is helpful to have a function that removes a pair (*k*,*v*) from a tree, where *k* is a key and *v* is a value. Your function will first search for *k*. When you remove *k*, you need to fix the resulting tree to ensure that it is still a binary search tree (and still a tree!).

(b) The amount of time it takes to search in a binary search tree depends on how far it is from the root to the node containing the search key. To speed up searches, it helps if the tree is as *balanced* as possible (that is, the longest distance to a node is as small as possible). Think about how you can create a balanced tree from a set of values, how you can rebalance an unbalanced tree, and how you can change addition and removal to restore balance.

Helpful tips

Loading a teachpack A teachpack is a file with Scheme definitions in it. Loading a teachpack means you can use any of the definitions as if they were built-in definitions. Under “Language” on the main menu bar, select “Add Teachpack...” The left column in the window that pops up is labeled “Preinstalled HtDP Teachpacks”. Choose “taxon.rkt” and click on “OK”.

Clearing all teachpacks Under “Language” on the menu bar, select “Clear All Teachpacks”.