15-122: Principles of Imperative Computation

Recitation 16 Solutions — Josh Zimmerman

Proof that `tree_insert`'s postcondition holds.

```
1  tree* tree_insert(tree* T, elem e)
2  //@requires is_ordtree(T);
3  //@requires e != NULL;
4  //@ensures is_ordtree(\result);
5  {
6     if (T == NULL) {
7        /* create new node and return it */
8        T = alloc(struct tree_node);
9        T->data = e;
10       T->left = NULL;
11       T->right = NULL;
12       return T;
13    }
14    int r = key_compare(elem_key(e), elem_key(T->data));
15    if (r == 0) {
16       T->data = e;  /* modify in place */
17    }
18    else if (r < 0) {
19       T->left = tree_insert(T->left, e);
20    }
21    else {
22       //@assert r > 0;
23       T->right = tree_insert(T->right, e);
24    }
25    return T;
26 }
```

To help you understand how `tree_insert` works, write a proof that it works.

As a reminder, when showing that recursive functions are correct, we first show partial correctness by first showing that in the base case the function is correct and then showing that if the function is correct on a smaller case it is correct on a larger case. We then show termination.

In this case, our steps will be as follows:

1. Show that the base case of the code (the case when `T == NULL`) is correct.
2. Show that if the recursive calls we make in the function (lines 19 and 23) are correct, then the function as a whole is correct.
3. Show that the function terminates on all input.

Solution: The preconditions for this function are actually not strong enough to prove its correctness!

This question was a homework question in F12 — see the writeup of the homework solutions, which you can find at. `http://www.cs.cmu.edu/~fp/courses/15122-f12/misc/hw-selected-solutions6.pdf`