15-122: Principles of Imperative Computation

Recitation 17

Josh Zimmerman, Arjun Hans

To understand Recursion, You Must First Understand Recursion

(a) Write a function `elem bst_max(bst B)` that returns the element with the maximum key in a given BST.

```
// Solution 1: iterative.
elem bst_max(bst B)
//@requires is_bst(B);
{
  tree curr = B->root;
  if (curr == NULL) return NULL;  // Empty tree.
  while (curr->right != NULL) {
    curr = curr->right;
  }
  return curr->data;
}

// Solution 2: recursive.
elem tree_max(tree T)
//@requires T != NULL;
{
  if (T->right == NULL) return T->data;
  return tree_max(T->right);
}
elem bst_max(bst B)
//@requires is_bst(B);
{
  if (B->root == NULL) return NULL;  // Empty tree.
  return tree_max(B->root);
}
```

(b) Write a function `int count_leaves(bst B)` that counts the number of leaves in a given BST.

```
int count_tree_leaves(tree T) {
  if (T == NULL) return 0;
  return count_tree_leaves(T->left) + count_tree_leaves(T->right) + 1;
}
int count_leaves(bst B)
//@requires is_bst(B);
{
  return count_tree_leaves(B->root);
}
```