Week 8: More Recursion and Exam Prep

Given that the professor has removed inheritance, polymorphism, generic types, and pointers from the topics covered in ENGR 151, there are fewer candidates for the programming problem on the next exam. The likely candidates will be objects and recursion, so this week and next week will focus more on those areas.

Last Week's Recursion

If you haven't done the recursion problems from week 7, see if you can get more progress. Do the Fibonacci problem as an introduction. Skip the Combinations problem and try the search problem.

For the search problem, if you code on a whiteboard, just output 1 if there is a solution through the maze and 0 if there isn't. If you code on your computer, please print the actual path if there is a solution through the maze or "impossible" if there isn't a solution.

Binary Search

Here is a classic question: What is the minimum number of words in a dictionary that you need to check before you locate the definition of a word *w*? You can assume that you know the number of pages in the dictionary is *n*. Try to figure it out on your own. If you aren't sure, we'll go over the process tonight.

Implement an iterative version of a binary search. Then implement a recursive version. Be careful - binary search is easy to mess up. Which version is more readable? Try this on the whiteboard if you'd like.

Superprime Rib

Please read the problem statement for Superprime Rib. The recursive solution to this is quite short. You should be able to code it out on a whiteboard assuming you have an isPrime function already provided. I can't think of any quick iterative algorithms off the top of my head that you might implement with ENGR 151 tools.

After you complete this problem on the whiteboard, feel free to code it up on your computer and I can submit it to the USACO autograder for you.

Binary Tree

Implementing a binary tree will be the perfect way to gain practice with classes and recursion. We will go over how a binary tree works tonight. The assignment will be to create a binary tree class so that a user can insert integers into a binary tree, check if a given integer is in the tree, and print the in-order, pre-order, and post-order traversal of the tree.

Note: Deleting from a binary tree is somewhat more complex, so we will not be going over that now. You can hear more about it when you take EECS 281.