Discussion Prompt Module 3
Brian Loughran
Data Structures

1. Describe a scenario where you see recursion at play in your daily life. What are some of the key features of this process or activity that make it recursive; reflect on the defining characteristics in the lecture. For each example, how is this different from an iterative process.
2. Describe a second and different scenario where you see recursion at play in your daily life. What are some of the key features of this process or activity that make it recursive; reflect on the defining characteristics in the lecture. For each example, how is this different from an iterative process.
3. Are there any scenarios you can envision in which recursion would really be impossible, that the process is so iterative that you could not look at it recursively at all?
4. What are some good, practical ways to organize your thoughts and ideas when working with recursion to manage the process a little more easily.

ANSWERS:

1. I spent most of my day today at work debugging ANSYS macros. This is an example of a recursive process. The basic structure of this process is to first check if the macro ran properly. If it did, we can move on to post-processing the results. If not, we have to "change something", then run it again. "Change something" is a pretty vague term, which can make debugging macros frustrating. There is potential for the debugging to go on to infinity if I can never solve the problem (fingers crossed this does not happen).

2. I also like to cook. When perfecting new recipes, you may consider that to be a recursive process. So the base case is to try the recipe for the first time. Then you eat it, and analyze the final product. Ask questions like: How does it taste, Is there too much salt, is the sauce thick enough, is the meat cooked to perfection, does it need more of ingredient x… Then you make changes to your cooking method, and recursively try again. The stopping case is vaguer in cooking than in debugging ANSYS macros. I guess I am always tinkering with the recipe a little. But a good stopping case is when you are comfortable enough to share it with friends and family. That usually is a final product you are proud of, and can be considered a stopping case to a recursion problem (cooking is not a perfect example of a recursion problem, but it is pretty decent)
.
3. Finding hobbies is a process which I could envision recursion being impossible. Hobbies are cool (my favorites currently are playing hockey and app development). But hobbies are also fickle things. They depend on who is around you, what your environment is, your health, your strengths, etc. Some hobbies which I used to like, but do not partake in anymore include poker (I recently moved, and no one I know plays poker) and woodworking (used to in high school, can't afford/ no space to create a whole woodshop). Finding a new hobby involves taking inventory of yourself and your surroundings and finding something that you like to do. There is also no stop case. Yourself and your environment will always be changing, which will always make certain activities more or less appealing. For example, if all the hockey rinks around me closed, and the closest league was 2 hours away, I would probably find a new hobby, as this would make hockey less appealing to me. Or one day I may grow old or out of shape, and be physically unable to play. This process is consistently iterative; therefore recursion is impossible to apply.

4. When tackling recursion problems, I like to first consider my end case, which is producing the solution of the recursion loop. I then consider my iterator. This is what tracks how many times I am going to recursively call this function. I make sure to manipulate my iterator so that it reaches the end case after the correct number of iterations. Only then do I add the "guts" of the recursion, the process which I want to put in place recursively. Once that is all done, a little testing and error handling is done to make sure it works and is hardened enough to handle all use cases, the recursive method is done and ready to go.