Recursion

Lesson plan and artifact created by Jennifer Yu

Recursion will take a few classes to accomplish. The physical artifact in this lesson is meant to visualize the call stack in recursion and help students understand fundamental concepts about how recursion works. The lesson is prepared for high school or college students taking an introductory Computer Science course that is nearing the end of its material.

Lesson #30: At-A-Glance

This lesson includes a physical artifact that will aid in helping students understand recursion in Computer Science. Produced in the form of a gif, the artifact represents the call stack and displays visually the concepts of building a stack of function calls, the idea of a large problem, which is then solved through smaller problems, such as the recursive and the base case.

Target Audience and Sample Situation

The audience the creator had in mind for this lesson is students who have started programming and are almost finished with their introductory course. This is normally when recursion is introduced, as the concept is rather difficult for novice programmers, but it is still important for computer scientists to learn.

A situation where this would be used is within a structured class lesson teaching recursion. The physical artifact is not meant to constitute all the teaching material, but to serve as a visual aid for students struggling to understand recursion.

Learning Objectives

By listening and actively engaging in the lesson students will:

- **Understand what a call stack is**
- **Understand the concepts of a recursive and base case**
- **Understand the idea of breaking down a larger problem into smaller constituents to solve**

CS Topics Covered

- **Recursion**
- **The Call Stack**
- **Functions**

Prior Knowledge Expected

- **Basic programming (has written programs before)**
- **Functions and how they are used**
- **Return statements and how they are used**
- **Basic mathematical concepts**

Recursion

Narration of the physical artifact:

First, there are no rings. This represents the empty call stack. The largest ring is then placed on and it represents the initial recursive function call. This is our “big” problem, one that can be broken down to be solved using the concept of recursion. This is physically manifested as the largest ring in the call stack. The rings themselves represent pending recursive function calls, waiting to be solved, which should be clear from the written narration. Afterwards, there is a slightly smaller ring, representing a recursive call. This ring is designated as a recursive case, which is a case that is slightly reduced from our original “large” problem, but still not solvable on its own. Again, another ring is placed on the call stack, representing this pending recursive call. Another recursive call is made in this visualization because it is later shown that the larger recursive cases derive their answers from the smaller recursive cases. The next ring is another recursive case call, so it is added onto the stack. Then, the final and smallest ring is placed on, which is the base case. This is designated as the smallest version of our big problem, a small problem we can solve. Since we can solve it, this ring is removed from the call stack (the wording says “popped” for proper terminology). One thing that is important to note verbally is that the small ring needs to be popped off before the larger rings, due to the physical structure of the ring stack; such is the case with the call stack as well. The smaller recursive case reduces to the base case, so after the base case is solved, the smaller recursive case can be solved as well. Similarly, the larger recursive case (second to largest ring) is popped off the stack after the smallest recursive case/ring is popped, since that ring was a reduction of the larger recursive case. In the end, we can pop our original large problem off the stack, since we solved the smaller subproblems through recursion. The call stack is then finally empty.