Recursion and Self-Reference

written by Lisa Wang

Suggested Section Plan (feel free to change as you wish):

1. *Start section with thought questions and thought candies.*
2. *Introduce students to the basic concept of a compiler. Make it mind-blowing.*
3. *Before students start working on the problem, go over data structures and passing by reference/by value.*
4. *Do the compiler program.*
    1. *Work out a strategy in pseudocode.*
    2. *Students work on the first problem in groups of two or three and write down their code collaboratively. It helps to check in with each group.*
    3. *Present possible solution and discuss robustness of the code.*
5. *Work on the challenge problem where students have think about how to write a self-reproducing program. Students can work in groups of two or three. The goal is not to write out the code in C++, but come up with a general strategy in pseudocode.*
6. *Talk about self-reference in other fields such as biology, chemistry, philosophy or any other related topic that you want to share with your sectionees.*

Crucial concepts:

- Use of data structures including maps and stacks.
- Passing by reference and passing by value.
- Recursion means to define something in terms of itself.
- In programming, we avoid infinite recursion by reducing the problem to a smaller sub-problem repeatedly until we reach the base case. Hence, it is not necessarily a paradox to define a function which calls itself.

Comments on the first problem related to compilers:

- It is very helpful to read Ken Thompson's speech "Reflections on Trusting Trust" (http://cm.bell-labs.com/who/ken/trust.html) that has examples of how the C compiler is written in C and can compile itself.
- You can give the students the code to extract the function name. It is on the back of the handout. Here is the detailed description of the function.

```
/* string funcName(string & input, int i)
 * --------------------------------------
 * Returns the function name of the parsed symbol by traversing the string
 * backwards until the beginning of the line by checking for the previous
 * newline character or the beginning of the file.
 * At the beginning of this function, i is the index of the opening curly
 * bracket.
 */
```

Comments on the quine challenge:

- The empty program is a quine, so applaud students who suggest that. Then, take it to the next level that the quine has to run at least one line of code.
- Reading in the file itself and printing it out is also a self-reproducing program, but encourage students to do it without file reading as a challenge.
- Make sure students understand the base case in the quine. E.g. the string in the program that contains every character in the program, but this string itsel. Otherwise, we would have an infinite cycle.
- Students are not expected to write a quine in actual code during section, but are encouraged to try it out later.
- To prepare for this section, try it yourself and code up a quine. It is a beautiful exercise!