# CS-102 Project 1 – A Strange Calculator

Professor: Brian Frost

Summer 2019

## Specifications

The purpose of this project is to test your ability to use the tools covered in your very first week of class. Although you have only been using the C programming language for a week, you already have access to a number of extremely important tools that require and deserve practive. As such, this assignment is a collection of small tasks to be performed using your three-lesson toolbox.

In short, I want you to create a "calculator", which can perform 9 tasks. When the program is run, the user should be prompted for a single-character input. Depending on this character, one of the tasks should be performed. The tasks are as follows:

- The input *h* (for *help*) should print out a small operation manual for the calculator. I would like to see a list of possible input characters and a brief description of their corresponding tasks. Try to format this nicely. Other than this task, you can choose which characters map where so long as each character maps to at most one task.
- A volume calculator for a sphere. Prompt the user for a radius (a float) and print the volume of the corresponding sphere. The number $\pi$ should be a constant, defined in either of the two ways discussed in class.
- An ASCII mapper. Prompt the user for an integer in the correct range, and print the associated ASCII character.
- A complex multiplier. Prompt the user for four numbers (floats or integers are fine) associated with the real and imaginary parts of two numbers. Compute the complex product of these two numbers using real multiplication and negation on these four numbers. Format the output as $a + bi * c + di = x + yi$ where the $a$, $b$, $c$, $d$, $x$ and $y$ are replaced by the inputs and outputs.
- An XOR gate. In class we covered the "or" operator, which returns "true" if and only if either of its two inputs are true. The XOR operator returns "true" if and only if exactly one of its inputs are true. Prompt the user for two integers, with the usual association of 0 with "false" and nonzero with "true". Print the output of the XOR operator applied to these two values. You should use your own logic, not a built-in C XOR.
- A vowel classifier. Prompt the user for a single character input. Use a "switch" statement to print one message if the letter is a vowel and another if the letter is not a vowel. Add a special case for the letter $y$ which is sometimes a vowel. I should be allowed to input either uppercase or lowercase letters and get the right answer.
- A square root calculator. Like we did in class, prompt the user for a float and take its square root using the math.h header. Remember you must add -lm when compiling.

- A reccomendation. Reccomend me a song and a piece of visual art (painting, sculpture, film) on two separate lines, but using a single print statement! Your taste will not affect your grade!
- Any ninth task of your choice in the spirit of the above – not just a print statement!

In general for these tasks, don't simply print the output – print a formatted string which contains the output. For example, if I want the volume of a sphere, I should get an output like "The volume of a sphere with radius ___ is approximately ___".

In general, try to add some statements to catch bad inputs. For example, make sure I can't input negative numbers for the square root function. However, don't go overboard – If I am prompted for an integer, I don't expect your code to be able to handle my inputting a string. This will not affect your correctness grade much, so don't spend too much time on it.

## Submission and Grading

Please submit your assignment as a .c file to b.frost@columbia.edu by midnight on Wednesday, July 17. Remember that lateness is penalized! Please state which operating system, text editor and compiler you used in your email. Also, please don't hesitate to ask as many questions as you need through email, or by coming to office hours.

Grading is broken down according to **Correctness** and **Style**. Your total grade is out of 100 points.

**Correctness (80 points)**: This is simply how well your program fits the exact above-stated specifications.

**Style (20 points)**: This is composed of efficient commenting, intuitive spacing and indentation, intuitive variable names, and overall elegant code. Unnecessary statements, poor placement of variable instantiations and otherwise difficult-to-read code will result in a loss of points here. Remeber to put a brief comment at the top of your code explaining the program's operation.