I started my construction of the calculator by writing a series of functions that take the argument of two integers joined by each operator respectively, evaluate them, and return the value. I then built a stack of integers named *evaluate* which also takes an argument of another stack of integers named *st*. *Evaluate* as a function first reads the top of the stack looking for a large number, unlikely to be input by the user, and assigns it to variable *x*. It then proceeds to remove *x* with the modifier *pop* and then goes down a sequence of *if* statements that are looking for one of five large numbers that will execute a sequence resulting in the evaluation of the two most recent integers. It starts with a similar process of assigning the top of the stack to a new variable and removing it, doing it again, and then checks if the stack is empty or not, either resulting in the two contained variables being evaluated. The process is repeatable through an *else* statement that runs if the stack still contains some element. Each statement ends with the returning of the evaluated element to *st* and will run until no element remains. Once the program reads the stack as empty, it runs *printf* to show the user the result of their inputted data and terminates. This is a fairly rudimentary program that could absolutely be improved on. The usage of large integers to check when the program should evaluate is an inherently flawed design due to it eliminating five possible numbers to be input. What is even more unfortunate is that if the program evaluates two integers into one of the five numbers, it will result in a runtime error that would need to be debugged. Furthermore, the user would be required to know that in order to evaluate the desired integers, they would need to input other integers which I imagine would be highly confusing to the uninformed, not to mention ridiculous. Regardless, the program still achieves the desired outcome of taking a stack of integers, evaluating them, and returning the value back to the user.