Using classes and structures to achieve the previously constructed calculator proved much more difficult due to the obfuscation and usage of multiple data types. While I was unable to correctly code the calculator, I do think that I have an understanding of the uses of obfuscation, classes, and structures. *Private* ensures that, once implemented, anything put into that block will be untouchable by those who do not have access to the *class*. This is useful for writing logic that should remain immutable as it is integral to the program's function. While in the header file, it will be unable to be referenced in *main* and must be mapped to something in *public* before it can be accessed. *Public* still resides within the class, but can be referenced in *main* or another part of the program and can reference things written in *private*. *Public* is useful as a middleman to map things from *private* to *main* while still upholding a level of obfuscation against the user. These added utilities made writing the program exponentially more difficult due to its usage of multiple "moving parts". It is, however, understandable in the case of no trust programming. Structures are particularly useful as they allow for complex data types. In the case of the calculator, they allowed there to be integers and characters in a container. Integers were naturally needed as I sought to evaluate them with an operator and characters would represent the said operators. This is a drastic improvement on the previous iteration which used *if* statements that looked for very large numbers to then run a sequence that resulted in the execution of an evaluation. This solved two house keeping issues with the previous program. The first being that the user would need to interpret the large numbers as the operators when inputting data into the stack and the second being that it represented an actual calculator more closely. After defining several key components in *private* and *public*, I mapped the two structures between spaces and began writing the function that would solve the stack in Reverse Polish Notation which is where I think I went wrong. This is where things became very confusing for me, whether it was calling the correct data types or using the proper syntax for the modifiers of my container. When compiled, the program quickly falls apart and I believe it to be due to the program calling functions that do not match, however I am unsure where. I did manage to get the modifier syntax correct which tells me that it is just a bug in the logic of the code. I did research online and any possible solutions that I found used a multitude of things that were not shown to me in the class and thus it felt dishonest to use them as I am sure I was given the necessary tools to write the program. Although the program's logic remains incorrect, I feel that my logic remains sound. I wanted to create a calculator with the use of complex data types and an object that would obfuscate the integral parts of the program while also containing a series of functions utilized by a larger function to take user input, put it into a container, evaluate the contents, and produce a result back to the user.

I intend to work on this program further as I detest leaving things unfinished, but I have to accept that I do not have the proper coding skill currently to make this work.