Object Calculator

In the hpp file of the code, the libraries that were included are, cstdio, stack, and vector. Cstdio was added into the header in order to include the printf function in the calculator. This allows for an actual output instead of receiving nothing back. Stack is included, in order to utilize stack. This allows us to have a set of integers and the ability to use all of them based on the way they were entered into the stack. Vectors were also included in order to utilize vectors. The vector is an array or set of numbers that can be used in the code, vectors are also pretty fast at what they do also. The code is using several different namespaces such as std::vector, std::printf, std::stack. This allows the code to look much cleaner and keeps the user from getting too clustered in order to read their code efficiently. We had a bunch of different problems that we needed to solve in order to get the code to just output a stack of integers and operations in order to add, subtract, multiply, divide. The main problem we faced was making a stack calculator using a monolithic object. We needed to obfuscate how the calculator does the evaluation. This meant using a class which utilizes public and private classes. The private class is unable to be seen by the user. This isn't hidden because we don't trust the user in this circumstance, but is being used because the user doesn't need to get into the class in order for the calculator to work. What we put in the private class was the stack that we utilized. Another problem was that we needed the calculator to actually do its job and solve equations we gave it. This didn't end up happening in the end but the path to get there is somewhat clear. We needed to set up something in our calculator that represented operations and numbers. We used a structure. This allows for us to house numbers for evaluation and use characters for numbers. In my code, I used the basic symbols for representing each of the five evaluations. "+" for addition, "-" for subtraction, "*" for multiplication and "/" for division. I also used "e" for evaluation, this is to be a fail safe if something unevaluatable is put into the code. The next problem we needed to address was what the basic operations of a calculator are in order for it to work. What we did for the operations was to define that only two numbers can be assessed at a time. We named these numbers "a" and "b", in order to represent any value given. Addition was set to a+b, subtraction a-b, multiplication a*b and division a/b. The "e" symbolizes that a number is ready for evaluation, but it is also a termination operation if the inputs are not correct. Another problem was to keep the structure's integrity. We used a dichotomy structure. This gave each number that was assigned a corresponding "e" which meant the number was ready for evaluation which was put on the bottom of the set. The operation was paired with an empty set or 0 to signify the operation in the dichotomy stack. Since the calculator is in reverse polish notation, the operation would be assessed first followed by the two numbers in the stack. This would give the computer something like +55, which would end up giving you an output of 10. We had a few unsolved problems too. These problems were, loading the stack in a proper order and also evaluating the full stack. I am unsure of how to solve these problems but would like to become better at programming in the future so I am able to solve them. For now, however, the problems go unsolved.