This assignment consisted of two program stitched together. The first program being an inventory manager from assignment 2. The purpose of this program is to allow groceries stores or businesses to keep track of their inventory in their store and to change the prices on the items based on sales. The second program is a dynamic storage. This is used to keep track of characteristics of different people such as their height, age, or hair color. This is good to keep track of different demographics of people. You can easily remove different categories or different names of different people. After putting these two programs together, the program will first ask the user if they want to use the inventory manager or if they want to use the dynamic storage manager. The user will then input either 1 or 2 depending on what they want to use. From there the program will go and execute the instructions for the program they specified.

When putting assignment 2 and assignment 3 together. This assignment quickly became a massive assignment that made it awfully hard to work with. Working with over 1,000 lines of code on one file was extremely hard. Breaking the code into a header files. One header file for assignment 2 and one header file for assignment 3 would have made the entire assignment a lot more manageable to work with and keeping my main file in the main C++ file. As I was frankensteing assignment 2 and 3 together I ran into a couple of interesting errors. Once I started implementing function prototypes, I got a least a dozen errors, so I decided to remove them. Another error I was running into was my main functions for assignment 2 and 3 could not find the functions. This was an easy fix as all it involved was reorganizing the functions so the functions were being predeclared and then the main function would call them.

The development of contracts removed the use of excessive error checking. Contracts are very handy when a user enters an invalid argument that could exploit a function. It automatically kills the function and tells the user why they are getting the error. Which is handy when you are trying to debug or exploit your own program.

The next part is logging. Logging is extremely useful in every possible scenario. The main debugging I like to use is simple print statements. If the program doesn't print out what you asked it to then its super easy to pin point the cause of error. This is easy in small program but once you start moving to massive programs where a bug could take a lot of steps and writing print statements all over again could become a lot and could be left in the code. Logging is handy because I makes the user feel more confident that the program is doing what it should. Like when Windows is updating but will not show what its doing after updating for 3 hours. Working in IT you see a lot of programs that will not give you any updates but will spin for hours and you wonder if the program crashed or broke. Logging is extremely useful for debugging and useful for the user to let them know where the program is when loading or what the program is doing.

Both of my program already include about a thousand preconditions to make sure the user enters the correct input. There's a couple of things that I could not check for such as NULL values. Contracts enabled me to check for these in a much easier way. The provided preconditions will check for almost everything I can think of. It will continuously prompt the

user for a valid input until the user provided the wanted input. Preconditions are nice as if the user were to make a mistake. It will allow them to try again and input the value or word they meant to input.

The development of tests was tricky for me. I have many different types of inputs that will ask for strings and numbers. I did try to test for everything I can. Writing looping statements to loop through and enter in A – Z or 1 – 1000+ to try to break my program. I have already implemented contracts before I started testing so when an invalid input was given. The contract will come into play and stop the program. This again helped out in the security of the program to prevent unwanted inputs from being provided. The development helped me understand all the weird combiniations that a user could put in. I know there's some pretty cool exploits you could do with SQL and gain access to any account you want without the password as long as you have a valid username and a quick google search for the needed string to break SQL. Testing for every possible combiniations of letters, special characters, and numbers will truly help you understand what will break your program and what is safe and from there you can add safeguards to help prevent your program from breaking from weird inputs.

This was a very fun assignment to work on. It was much easier than all the other programs as I didn't have to come up with unique idea from scratch and code it from zero. All I had to do was put two programs I have already made and add in a couple extra lines of code such as contracts for all my inputs. I also had to implement logging which is super useful for telling the user exactly what is happening in the program as it runs. Also having to write test cases to check for contract violations and see what will violate the contract and what wont and having to adjust the contract to make it all work properly. This assignement deepened my knowledge of all the different parts of C++ to help me gain a better grasp on the language. I also feel like this will be exceptionally useful for the final that is coming up soon.