

Figure 1 – Output of Addition Test

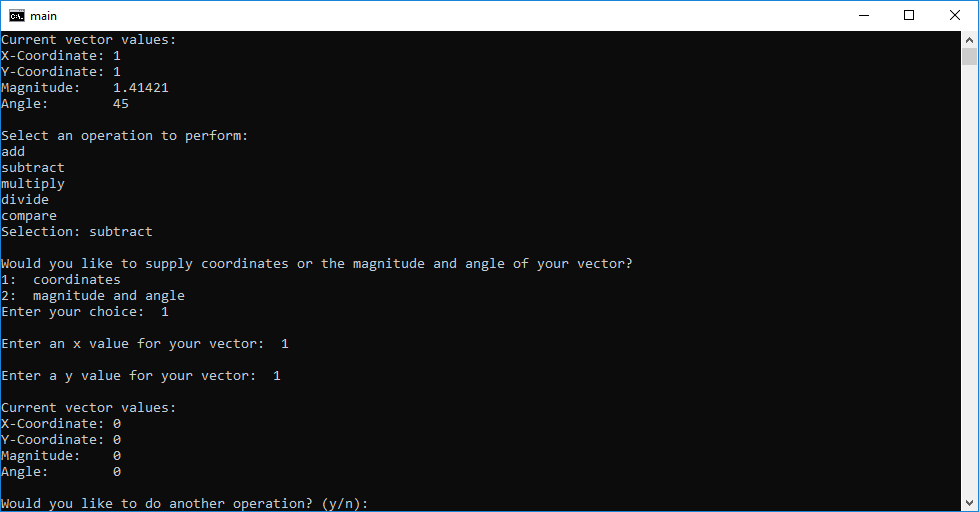


Figure 2 – Output of Subtraction Test

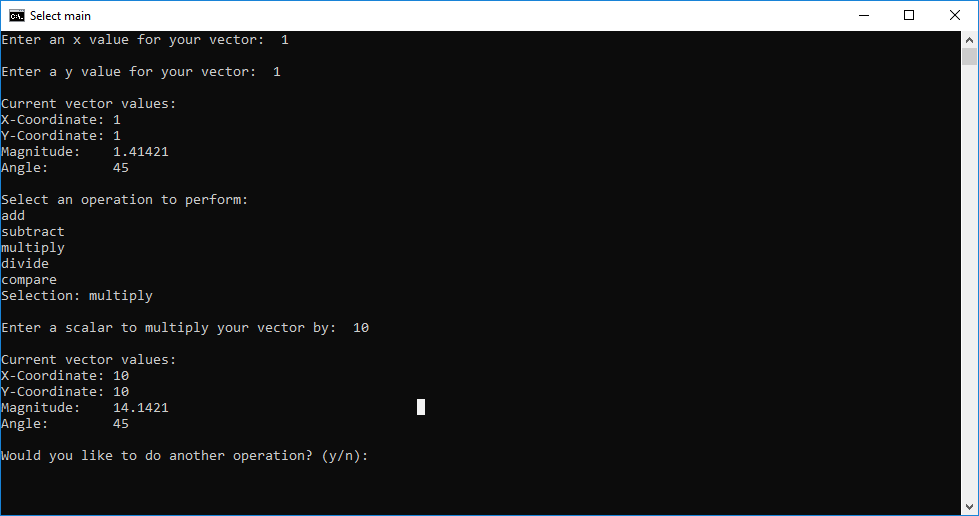


Figure 3 – Output of Multiplication Test

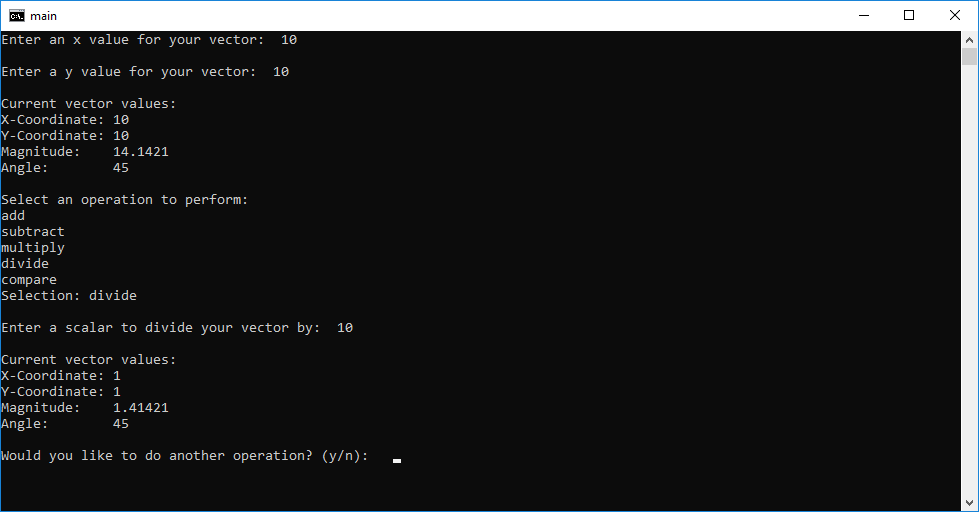


Figure 4 – Output of Division Test

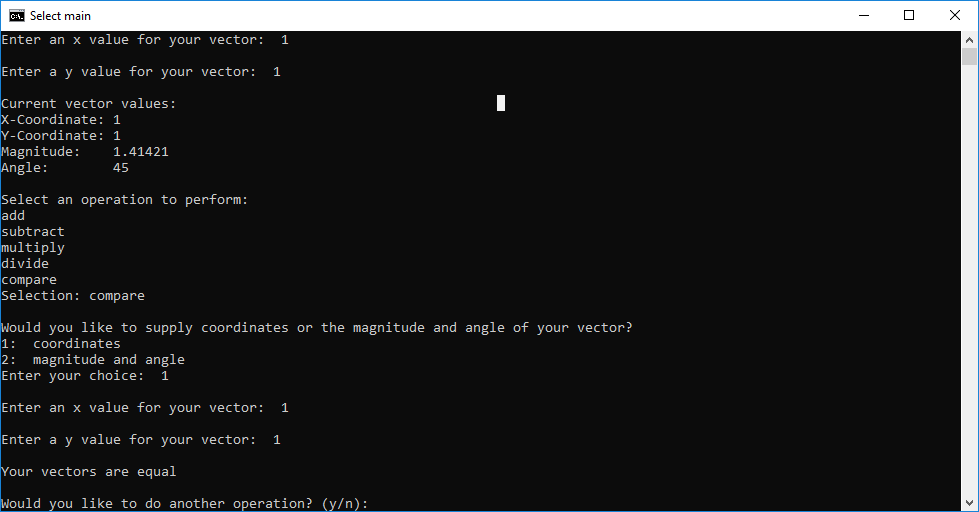


Figure 5 – Output of Comparison Test

In this assignment, the goal was to create a class and implement operator overloading. In the class, there are several member variables, functions, and constructors. The member variables, x, y, magnitude, and angle were all kept private. The reasoning behind this is that any change to one will cause changes to the other. This means it would be a bad idea to allow the values to be changed directly. This is why in the setters, math is done to adjust the other values accordingly when you make a change to 1 value. As far as other basic members go, there is a print function to output vectors, and also getters for each of the values. These are needed for calculations in the operator overloading section.

Three constructors were provided in the class. The default constructor and the 2nd constructor which accepts an x and y coordinate are very similar. While the default sets the values to zero, they both make use of the setter functions from earlier to ensure that the magnitude and angle are calculated properly when an instance is made. The last constructor accepts a magnitude and angle. This constructor can only be used in conjunction with the user supplied vectors that are created after the initial vector. For these vectors, the user can choose rather they want to supply coordinates or magnitude and angle. Should they select magnitude and angle, the constructor does the appropriate math to calculate the x and y values.

No modifications or new members were added to the class to implement it into the main or complete the operator overloading. The class was set up well to begin with due to proper planning. When looking at the requirements, the first thing that was done before any code was written was to research all the equations needed to do the vector math present and how they could be implemented in C++. Luckily this ended up being rather straightforward as “<cmath>” has all the trigonometric functions needed. While it is probably rare for a class to be created initially with all the members that will be needed, the plan we had and the way we went about coding is generally a good idea. The most important considerations are to figure out what you will be doing with the class, what helper functions may be useful, what problems may pop up ahead of time, and if there are any standard libraries that can be used. These considerations will generally keep any problems that pop up to a minimum compared to jumping straight to coding.

Most work on the main was done by Zach Hammitt. Most of the work on the class definition and declaration was done by Dustin Seger. The work was divided evenly.

All files should be compiled in the same folder with default windows settings for the gcc compiler. Lab03.cpp contains the main