**original.cpp**

Within the program original.cpp, I tested multiple trials of numbers. In the first trial, I didn’t do anything weird or input anything with nonsensical numbers. When asked “How many registered voters were surveyed?” I entered 20. When asked “How many of them say they will vote for Gavin?”, I entered 15; When asked, “How many of them will vote for Brian?” I entered 5.

This gave an output of “75% say they will vote for Gavin. 25% say they will vote for Brian. Gavin is predicted to win the election.” This trial is logical and all the calculations add up.

In the second trial of inputs, I ended up inputting numbers that doesn’t really make sense. When asked “How many registered voters were surveyed?” I entered 1. When asked “How many of them say they will vote for Gavin?”, I entered 19; When asked, “How many of them will vote for Brian?” I entered 20. This gave an output of “1900.0% say they will vote for Gavin. 2000.0% say they will vote for Brian. The program still went through and was able to build but logically it doesn’t make any sense. If only one person was surveyed, how is it possible that 19 individuals said they were going to vote for Gavin while 20 other individuals said they were going to vote for Brian? The percentages also add up to way above 100%.

In the third trial of inputs, I ended up, once again, inputting numbers that doesn’t really make sense. When asked “How many registered voters were surveyed?” I entered 0. When asked “How many of them say they will vote for Gavin?”, I entered 32; When asked, “How many of them will vote for Brian?” I entered 20. This gave an output of “inf% say they will vote for Gavin. inf% say they will vote for Brian. This glitches because one of the lines of the program divides the votes of an individual by the number surveyed and if there were 0 individuals surveyed, that would cause an error. You can’t divide by 0.

**logic\_error.cpp**

To cause a logic error within this code, I decided to switch the > to a < on line 32 of logic\_error.cpp. ( changing if (forGavin < forBrian) to if (forGavin > forBrian)) Whenever Gavin has fewer people saying that they will vote for him, it’ll output “Gavin is predicted to win the election” when in actuality it should be “Brian is predicted to win the election”. The same goes vice versa. The code will still build, it just won’t give the right result.

**compile\_error.cpp**

One of the two errors that I introduced into the compile\_error.cpp would be a semi-colon syntax error. On line 26, I removed a semi-colon from cout<<endl. This causes the compiler to not be able to finish that line as semi-colons are indicators that the line has ended. Since there’s no semi-colon at the end of that line, the code will not be able to execute. The second error I introduced would be on line 10. I switched int forGavin to string forGavin. This causes the code to crash because the input that we enter for forGavin would be an integer and the operators associated with this variable also needs it to be an integer. For example, forGavin is multiplied by 100 on line 20 but this doesn’t make sense because a string variable can’t be multiplied. This will cause the code to not execute.