# Input Provided Under Step 5

1. When I input 10 for the number of registered voters who were surveyed, 5 for those who said would vote for Newsom, and 5 for those who said would vote for Cox, the output says that ‘Cox is predicted to win the election’, even though it is a tie. This is because there is no tie condition present in the code. The way the code is written, as long as the number of registered voters who said they would vote for Cox is *not less than* those who said they would vote for Newsom, the program outputs that Cox is predicted to win the election, even if it actually is a tie.
2. When I input 10 for the number of registered voters who were surveyed, 12 for those who said would vote for Newsom, and 13 for those who said would vote for Cox, the program runs smoothly, but outputs that ‘120.0% say they will vote for Newsom’ and ‘130.0% say they will vote for Cox’. This is clearly nonsensical input, but the program still accepts it and runs the program successfully, because the program doesn’t check to make sure that the sum of the number of voters who said they would vote for Newsom and those who said they would vote for Cox doesn’t exceed the number of registered voters who were surveyed.
3. When I input 2147483648 for the number of registered voters who were surveyed, the program fails to continue to run as intended. Instead of taking further input from the user, it directly outputs ‘0.0% say they will vote for Newsom’, ‘-12.7% say they will vote for Cox’, and ‘Newsom is predicted to win the election’. This error occurs because the variable used to store the number of registered voters surveyed, numberSurveyed, is of the ‘int’ type, so it has a maximum value of 2147483647. Since the input value exceeds this value, the program fails to proceed as intended. Similarly, when I input 2147483648 for the number of voters who said they would vote for Newsom, the program again fails to continue to run as intended. However, quite curiously, when 2147483648 is inputted for the number of voters who said they would vote for Cox, the program proceeds to run smoothly. This is because, for reasons unknown to me, the computer automatically caps the value for the number of voters who said they would vote for Cox at 2147483647. I tested this theory by inputting 1 for the number of registered voters who were surveyed, and 3000000000 for those who said they would vote for Cox, and the output displays ‘214748364700.0% say they will vote for Cox’, implying that 2147483647 was entered as the input value for the number of voters who said they would vote for Cox.
4. When I input 0 for the number of registered voters who were surveyed, 1 for those who said would vote for Newsom, and -1 for those who said would vote for Cox, the output says ‘inf% say they will vote for Newsom’, ‘-inf% say they will vote for Cox’, and ‘Newsom is predicted to win the election’. Although the program proceeds to run successfully, having outputs of ‘inf%’ and ‘-inf%’ seems nonsensical. This is because it is not possible to compute the division of a number by 0, so the compiler is programmed to recognize when a number is being divided by 0 and output ‘inf’ when the number is positive and ‘-inf’ when the number is negative. Similarly, when I input 0 for the number of registered voters who were surveyed and for both the number of voters who said they would vote for Newsom and those who said they would vote for Cox, the output says ‘nan% say they will vote for Newsom’ and ‘nan% say they will vote for Cox’. ‘nan’ stands for not a number and is used to represent undefined values.
5. When I input -10 for the number of registered voters who were surveyed, -3 for those who said would vote for Newsom, and -7 for those who said would vote for Cox, the output says ’30.0% say they will vote for Newsom’, ’70.0% say they will vote for Cox’, and ‘Newsom is predicted to win the election’. If we ignore the fact that we inputted negative values, the program seems to be contradicting itself, because even though it states that a greater percentage of voters say they will vote for Cox, it predicts that Newsom will win the election. This is because, when predicting whether Newsom or Cox will win the election, the program compares the number of voters who said they would vote for each candidate, not the percentage of voters who said they would vote for each candidate. Since -3 is greater than -7, the program predicts that Newsom will win the election.

# Errors Introduced Into logic\_error.cpp

1. In the formula for calculating the value of the variable pctNewsom, I entered ‘1000 \* forNewsom’ instead of ‘100.0 \* forNewsom’. This will cause the output to display 10 times the actual percentage of voters who said they would vote for Newsom. However, this change wouldn’t have any effect on the prediction of who wins the election, because that prediction is done by comparing the variables forNewsom and forCox, not pctNewsom and pctCox.
2. In the section of the code dedicated to predicting the winner of the election, I entered ‘if (forNewsom < forCox)’ instead of ‘if (forNewsom > forCox)’. This will cause the program to output the wrong candidate as the predicted winner, because it will output ‘Newsom is predicted to win the election’ if forNewsom is less than forCox, and ‘Cox is predicted to win the election’ is forNewsom is greater than or equal to forCox.

# Errors Introduced Into compile\_error.cpp

1. In the beginning of the code for this program, I ‘forgot’ to enter ‘#include <iostream>’, which is the part of the code that allows me to get input from the computer’s input devices (the keyboard in this case) and output data to the computer’s output devices (the screen in this case). The compiler is therefore unable to process the following identifiers in my code: ‘cout’, ‘cin’, and ‘endl’. The compiler therefore is unable to compile the program and generates the error message ‘Use of undeclared identifier’ on every line that I have used one of those identifiers.
2. On line 30 of my code, I did not end the line with a semicolon (;). This causes the compiler to fail to compile the code, and the compiler displays the error message ‘Expected ‘;’ after expression’ on line 30.