Critical Thinking 4

Gregory Norris

CSC 450

Colorado State University Global Campus

Critical Thinking 4

The goal of the Critical Thinking 4 assignment is to create a C++ program which takes user input for the number of standard hours, pay rate and overtime hours, then calculate and return the salary for the information given. In addition to this calculation, the output is formatted using the setw() and setprecision(). In addition to taking input from the user and returning a formatted string, the program should also create a pointer for each variable and print information regarding the variable’s destination address, dereference and pointer address.

Since security is the primary focus of this course, I focused first on sanitizing the user input and attempting to catch common errors which would interrupt the program’s intended processes. I choose to use cin and cout predominately, since they have fairly robust integrated error handling and are capable of handling most user inputs with minimal extra coding. I wrapped the user input statements in a try statement to catch out of range exceptions should the user input be too large to process. After obtaining the required input, the cin is cleared and the ignore() function is used to prevent any subsequent information from being processed by the program. This is to override a behavior where the user can insert multiple arguments on a single line and have each one assigned to variables in the program sequentially.

To best sanitize the input, I decided to take the user information as a string first, then use stod() to parse the string into a type double. I chose to do this rather than outputting cin directly into a double since it was easier to catch errors regarding the stod() process than to catch errors from piping cin directly into a double variable. I also qualified the input to be within what I would consider sane bounds, 0 <= I <= 999,999,999, since I found that overloading the stod() function with too large of a value caused it to return either a truncated or incorrect value. Assuming that the program has produced a viable double value at this point, I believed that I could safely assign pointer values without requiring more additional checks.

To output the program, I used cout. I saved the initial cout parameters at the beginning of the program so that I could reset the formatting for printing the pointer values, since using setw() and setprecision() was rounding off the pointer value addresses or displaying them in scientific notation, which was not desirable. I stored the cin flag state as an ios value, which was later applied to cin to undo the formatting changes made to display the program output.

Overall, the program is fairly secure and handled all of the test-cases I threw at it. I am sure that an experienced programmer could find ways to break this program, although it seems to do a good job of handling most inputs.