Zackery Devers

Peter Snipes

12/3/21

Cmpsc 200

Calculator

Team 7

The project we proposed consists of a basic calculator with simple functions that allows a C program to be executed simply. In our calculator we chose to include the plus, minus, multiply, divide, exponent, and log function. The challenge we had to overcome was making the calculations work correctly without the use of the math.h library function. Unfortunately we could not use this library for our code. The reason that the library function could not be used is because the library function contains the built in log and exponent function. In order to challenge ourselves when making a program that runs using the plus, minus, multiply, divide, exponent, and log functions. Using the math.h library function was not an option. This is because if we utilized other functions it would better showcase our understanding and knowledge that we acquired in class while also showcasing what was learned during our further research of C. Successfully completing a challenge was ultimately the better route considering we wanted to showcase our newfound knowledge.

To start off our code we included the stdio.h library for basic functions. We initialized the main function and also declared our operator terms. The op variable is going to stand for all of the possible switch cases that the user could choose from and that will determine which function you want to calculate. The next step is to then declare the two variables that the user picked which will begin the mathematical process. Additionally, we decided to double the first and second inputs which will allow the user to input numbers with decimals as well as numbers without. After that we made a long double with the name of the result that equals 1.0. This will allow the result to be multiplied by the first input for the exponent function. For the next line of code we made a print statement that explains the first step of the calculator. The calculator operation will be inputted alongside the operand and whatever the user decided to choose will be scanned and stored into the memory as well. Then from the information that was scanned into the memory it will be saved as the op variable. Following this step our code is found to be outputted with a print statement. Located inside of the print statement are the two operands that are needed to complete this function. The two operands are then scanned and saved into the first and second variables which are needed to run through the switch cases.

In order to get this code to work we decided the only way feasible with what our knowledge consists of is by using the switch cases for each separate mathematical function. To figure out what switch case to use we set up the op variable, which ultimately helped us determine which switch case made the most sense in using. Another option that is capable within our code is dependent on what the user wants to do. If the user wanted to use addition they would then have to input + when the command line text prompted an operator. The additional coding that was done was straightforward. Straightforward being, it was built in basic functions of C

making it simple for what our standards of difficulty with mathematical operations are. These standards come from what we have learned in class and from further research we have conducted during this school year. Surprisingly the only steps we had to take were in order to implement the addition of first and second, and print it in the same line. The same steps were used for the subtraction, multiplication and division functions. We had to get the exponential function to work in order to accomplish this step we created a switch case that has a while loop embedded. To make this embedded while loop work we had to implement codes and make sure the second value was not a zero. We had to do this because we can not raise a number to the power 0. If a number was raised to the power of 0 we implemented the code to go to the next line. The results ended up being taken and stored as the first value. After the results were stored as the first value the second repeated until the value came down to 0. Continuing this, repeat the prior process until the correct value is found. Concluding to the while loop ending and the new result being printed out.

Unexpected challenges we faced as a team were time management and implementation of code for the log function. log is challenging because we kept getting unexpected errors with the code we put into our file. Time management was an issue because doing this alongside other technical assignments that were given to us proved to be challenging. It was challenging to give all of our effort to one assignment when there were multiple to complete. Not having the capacity to channel in on one assignment restricted us from giving uninterrupted focus to a specific one. Which would have allowed higher performance. However through setting aside time to complete each assignment we managed to get done what was asked of us and have a performance we are

satisfied with. Log function was difficult because we are still struggling to find the desired results. Each attempt we have tried simply is not giving us the data we would like.

Changing our initial model framework or project skeleton code was an issue of concern. We did not end up having to change either the model framework or the project skeleton code. We decided to choose between the two project tasks, project task 1 and project task 3. In project task 1 we were required to solve one or more low-end computing problems, programmatically using either C programming language or MIPS or both. A few examples of what we did so far in this direction, is to implement the shift, binary addition, and subtraction operators, converters such as binary to decimal and decimal to binary. In this track, it is important to focus on low-end computing tasks and simulate the execution of those tasks through one or more programs implemented in C and/or MIPS. Additionally in project task 3 we were required to develop an idea for their own project that focuses on one or more real-world topics in the field of low end computing. In this track, if you had chosen to implement in C, you are expected to use Dynamic and Heterogeneous Data Stores such as Pointers, and Structs in your implementation. Ultimately we decided that project task 1 was the best fit to accurately showcase the things that we learned in class. What we learned in class that is similar to project task 1 was the concept of low-end programming computer tools when implementing with C. Low-end programming computer tools are to implement the shift, binary addition, and subtraction operators, converters such as binary to decimal and decimal to binary. These are all different types of low-end programming computer tools. Which we also learned about in class and through further exploration when completing various other assignments based on this criteria. In class we were also taught about mathematical operations such as: plus, minus, multiply, divide, exponent, and log functions. All

of these factors prove to be more useful than the skill set project task 3 provides. Also in project task 1 we can use outside resources to complete a technical assignment which ends up being very useful.

With more time we will be able to complete the log function. We are in the process of fixing the crashes that happened when we initially ran our code. Then continuing to add on the required source code to aid in fixing the faults in the program and making said program run more efficiently. The reason the log function is still being worked on at the moment is because there were some complications with completing it. Thus leading to there needing to be more time dedicated to finding a solution. Which is currently what is going on right now. This is a step in the assignment that has been most complicated for us to finish. We have attempted a couple different options yet they still are not exceeding our expectations of the level of work we deem satisfying. Satisfying being meeting the correct outcome. The default we have concluded on is for if they put an incorrect operator. It is a print statement that explains that there is an error and that the operator is not correct.