# PROBLEM DEFINITION

This program allows a user to perform simple math operations and then check the answers to these operations. The user picks which math operation to perform. The user then enters the operands and the answer. The program then checks the answer and notifies the user if it is correct and what the correct answer is if it is incorrect.

# ANALYSIS

The variables I used to hold input data from the user were menu\_selection, num\_1, num\_2, and user\_answer. The menu\_selection variable stores the option selected by the user at the main menu and is a string. The num\_1 and num\_2 variables are both integers that the user inputs to perform the mathematical operation on. The user\_answer stores the users attempt at solving the chosen operation. The user\_answer variable can be either a float or an int. This data type depends on the operation the user chose. The float is for division, and an int is used for all other operations. I have used the calc\_result variable to hold the answer after the program performs the mathematical operation. This number will be an int for all operations except division which will result in a float. I then round this float so it can be compared to the user\_answer which should be to 2 decimal places. The calc\_result is calculated using the following formulas:

Addition - calc\_result = num\_1 + num\_2

Subtraction – calc\_result = num\_1 – num\_2

Multiplication – calc\_result = num\_1 \* num\_3

Division - calc\_result = num\_1 / num\_2 then rounded to 2 decimal places

# DESIGN

1. Display the menu for the user and get the first user choice
2. Repeat the following steps if the user choice isn’t quit
3. Test the choice to see if it is division because division by 0 isn’t possible
4. Prompt the user for two operands
   1. If the choice is division repeat the following steps while the answer is 0
5. Prompt the user for two valid numbers
6. Test the choice for division for proper user answer prompt to ensure the user enters the answer to 2 decimal places which the program will compare to it’s own answer
7. Get the users answer
8. Calculate the answer for the menu choice with the given operands
9. Test the answer with the users answer
10. Tell the user whether they got the problem correct or if they got it incorrect
11. If they got it incorrect, tell the user the correct answer
12. Get the users menu choice again
13. After the loop finishes, thank the user for using the program.

# IMPLEMENTATION

I created my implementation of the math tutor program in Visual Studio Code. VSCode is running on my custom-built pc with a Ryzen 1700 3.65GHz CPU and Windows 10. I also worked on the program using Linux Mint 19.2 on the same hardware using VSCode. I ran the code with the built-in terminal in VSCode.

I started out by reading the program specifications to see exactly what my program needed to accomplish. Initially I planned out my program by placing comments in the empty document. I used these for all the components that were required. I didn’t write any code until I had a good amount of comments laying out what my program would look like at the end. While I was planning on using the while loop to control the menu, I wanted to get an initial value for the menu selection. I did this by printing out the menu and prompting the user before entering the menu loop for the first time.

Once I had the user’s selection, I moved into the while loop that would run until the user entered the quit command. The first time I wrote out the code that simply went through the steps of prompting the user for the operands and the answer. After I had those values, I calculated the value of the operation that was to be compared to the user’s answer. I implemented the test for the answer comparison and displaying the outcome to the user. This gave me a basic program that ran a couple of basic tests successfully.

The next step I took was to read the user stories provided that gave me an idea of what a typical user should see when using my program. This reminded me that the division operation was going to need some input verification for it to work. I decided to use an if/else statement to filter out the division option from the user. This allowed me to create a small loop to ensure that the user didn’t enter a 0 as the second number. This would cause a division by zero which is invalid. The same user story also indicated the answer needed to be to 2 decimal places. I implemented another if/else statement to check for division when prompting for the user’s answer to inform them that the answer must be to 2 decimal places. I altered my calculation of the division also to use the round function to set it to 2 decimal places. This provided similar user answers and program results to compare.

Following the logic, I tried to make the user menu look a little nicer. I also added spaces between the program outputs to make it easier to read. I made sure the final output, showing if the user had gotten the answer correct, stood out by surrounding it with some symbols.

The first tests I ran on my program were just simple addition tests. After these initial tests, I used the values given in the specifications as some test values. Following these tests, I tested the capitalization of each menu option along with an incorrect and correct result. I made sure to test my cases that I had to make special exceptions for including dividing my zero and testing the result as a float. Some test data included:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **First number** | **Second Number** | **User answer** |
| Addition | 33 | 3 | 35 |
| addition | 5 | 4 | 9 |
| addition | 8 | 12 | 20 |
| Subtraction | 5 | 10 | -5 |
| subtraction | 2 | 3 | 1 |
| subtraction | 35 | 18 | 17 |
| Multiplication | 4 | 3 | 12 |
| multiplication | 5 | 10 | 49 |
| multiplication | 25 | 4 | 100 |
| Division | 10 | 0 | n/a |
| division | 1 | 3 | 0.33 |
| Division | 5 | 2 | 2.5000 |
| division | 1 | 2 | 0.75 |
| Division | 3 | 4 | 0.75 |
| quit | n/a | n/a | n/a |
| Quit | n/a | n/a | n/a |

I had a few problems when testing my program. The biggest problem is that the program can not handle the case where a user inputs a data type that it doesn’t expect. The program expects two integers to perform all the calculations, but if the user enters a string or float the program will crash. I added a feature that if the user enters a string that isn’t on the menu it will restart the loop when the program gets to the test for the calculations.