CSE 115 Section 2

Group Project (Number 9)

Group Members	ID Number	Email Address
Kazi Abu Baqr Khalil	2511608042	kazi.khalil.251@northsouth.edu
Sidratul Muntaha Rahman Khan	2514009642	sidratul.khan.251@northsouth.edu
Prothom Khondker	2512647042	prothom.khondker.251@northsouth.edu
Shakibul Alam Ashik	2514440043	shakibul.ashik.251@northsouth.edu

In this project, we will be using the C programming language to create a code that, when built and run, displays the basics of a calculator on the console.

This code mainly consists of 3 parts:

- User Interface (UI) Design
- Expression Handling
- Expression Evaluation

Designing the interface:

Firstly, we have to include <conio.h>. This will specifically help us in interacting with the console/output display which will ultimately aid in inputting different numbers and functions for mathematical calculations.

By using <conio.h>, we are able to use commands such as:

getch() - This pauses a program's execution until a key is pressed to allow the program to progress.

clrscr() - This will allow a user to refresh the calculator's output screen and enable them to input again.

gotoxy() - This moves the cursor to the desired position written within the code, so that the printf() function will always show the output in the same position.

textcolor() - Simply put, this changes the color of the text. Either put the name of the color within the first brackets or use an online chart that assigns specific colors to numbers.

textbackground() - Similar to textcolor(), this will change the background color of the text.

cprintf() - Just as when we use printf() to output the result of a code on the console, this function will print the formatted and colored version of the text in the console.

Next, by using these functions we can build the basic layout of a calculator. To do this, you will need to build multiple functions with variables that will position the different numbers on the calculator.

To place borders around each ‘button’ in the calculator shown in the console, you will need to place ASCII values in variables and run loops. Moreover, some borders’ sizes can be altered to align with the console output.

The color of these borders can also be set. While on the topic of colors, a specific conditional statement is set to change the color of the buttons of the calculator for a fraction of a second when they are pressed.

Expression Handling:

This explains how the numbers and operators pressed are used and stored in arrays to be used later on and also how different errors can be eliminated.

Here, we introduce an array to store whatever the user types in the calculator until ‘enter’ is pressed. This is done using a do-while loop. We should be careful while doing this as we have to increment it to add multiple characters.

Digit Checking - We have to place a conditional statement to ensure that only numbers and operators can be used as inputs in the console.

Backspace/Delete - We also have to set a loop with a decrement to allow the user to delete the last entered character whenever they wish to do so.

Decimal Point [.] - This is to check that a decimal point is only placed once in a number, so if multiple decimals are placed, an error shows up on the screen.

Operators - Undoubtedly the most important aspect of solving mathematical problems, we have to use switch case and conditional statements in order to carry out different operations, such as addition, subtraction, multiplication and division.

Incomplete expressions - If someone enters an operator but misses out a number before or after it, the program should produce an error message.

Click effect of buttons - As discussed previously, when the user clicks on a character in the calculator, the same character on the screen should have a different colored border for a fraction of a second. To accomplish this, another function is to be created and carried out, which contains a switch-case and delay() in between so the light-up click effect is visible on the console.

Expression Evaluation:

Finally, to obtain the results of the user’s inputs and calculations, we will have to get an output shown on the calculator screen. In order to carry this out, we will need to first create an array where the answer will be stored. This will then be printed onto the screen at the same place where the original cursor was placed at, hence we will need to use the gotoxy() function again.

For the convenience of the user, we would have to place a loop so that when a task is successfully completed, the user can simply press another character on the calculator to prompt another task.

Video Reference Links:
https://youtu.be/mnQrRBawsok?si=mWlExWUJGzdciBsb
https://youtu.be/QGO2IDsAN9g?si=haqTsKGKe0azQO-W