In this lab, we program two subroutines double_in that prompts for a value from the keypad and printf_lcd that prints the string passed to the LCD

1. Description

There are three major functions in main.c for this assignment. The first function is main(), where it calls another function double_in twice and assigns it to two double parameters which is ultimately printed to the LCD by calling printf_lcd. The second function is double_in, which prompts a valid decimal value from the keypad and prints it to the LCD. Lastly, printf_lcd accepts a string and prints it to the LCD screen, it works similarly with the printf command.

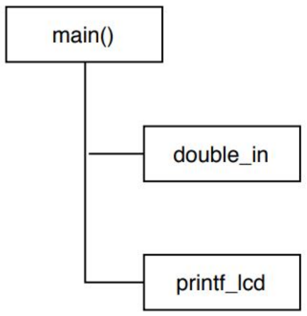

Figure 1: Function Hierarchy

2. Testing

To test double_in, we would need to call the function in main while passing a desired string to the function. Next, main should print the value returned by the function simply by using printf and checking the console. The function double_in should print the prompt passed to the LCD, and it should catch 3 types of errors, and reprompts the user:

- If the user inputs nothing and presses enter, the LCD should show: "Short. Try Again."
- If the user inputs the UP or DOWN key, the LCD should show: "Bad Key. Try Again."
- If the user inputs more than one negative sign, the LCD should show: "Bad Key. Try Again."
- If the user inputs a segment of .. (double periods), the LCD should show: "Bad Key. Try Again."

Once the function is executed, check the value printed in the console. The value should be the same as the value returned by the function double_in.

To test printf_lcd, call the function in main and pass a desired string. The LCD should show exactly the string passed to the function. To test the length of the string it should return, call the function and assign it to an integer parameter, and use printf to print the integer on the console. The value printed should be the length of the string passed. If there is an error encountered, then the value returned should be -1.

3. Results

Briefly discuss the results of your experiment. State how successfully the program runs, noting any unsolved problems. Answer any specific questions suggested in the assignment. Include the results (plots, etc.) of any required analysis. Suggest possible improvements, such as extensions to the program beyond what is required, that might be made with more time.

In my main program, I called double_in twice and assigned each to two parameters and printed them to the LCD using printf_lcd. Upon the first function call of double_in, the LCD prints the prompt passed to the function, which validates the function. When I press enter, the LCD will show "Short, Try Again" on the second line. I added a code that prints spaces on the first line to clear the first line so that the bad key pressed is deleted after we press enter. When I added an UP, DOWN, repeating "..", and "-" sign beyond the first index, the LCD will show "Bad Key. Try Again" on the second line.

Finally, when I pass a legal value, the LCD will print the next prompt which corresponds to the next double_in call. The same behavior is repeated, which is a success. After a legal value is passed, main will print the two values assigned to the LCD, which is the expected behavior.

To check printf_lcd, I called printf_lcd in main, and passed strings of different lengths. First, I tried passing the string "four", and I printed the value using printf, which yields 4 (the correct length). Second, I tried passing the maximum length, which is 80 characters (since the last character is reserved for EOF), and 80 is printed in the console. Lastly, I passed a string of more than 80 characters and -1 is printed in the console, which means that there is an error in printing all the strings (which validates the code since the char buffer's length is 80). Therefore, my code is validated.

One possible suggestion for this assignment is to extend the definition of illegal value to double_in. Currently, if there is a point (".") scattered in two different places, the function does not notify that there is an error. This bug is a potential area of improvement. If we are given more time, I would carefully design the parameters more carefully so that the code runs smoother. Examples of this would be removing static int iterators, using try catch control structure to detect the error, and assigning repeating variables such as the warning "Bad Key. Try Again." to a specific variable.