Things I learnt in MS2:

First and foremost, I learned what is an input buffer and how to clear the input buffer. An, input buffer is a data that was left in the buffer after the scanf function accepted the necessary input and left the rest in the buffer. The next time when a scanf is called, it will automatically accept that buffer without waiting for the user to enter the values, which is why it is very important to clear the buffer after every scanf if the characters entered are to be greater than the size of the maximum characters allowed in a string. The clearKeyboard function whenever called, runs a loop to flush all the input buffer left until it reaches a newline character at which point it returns the control to the caller of the function.

Furthermore, I also learned how to call and pass arguments to a function. Functions are made up of, 1. Function Prototype or Function Declaration, which specifies what a function will look like, which includes its return type if any, function name and its parameters along with their types, 2. Function Definition, which means where the function is defined along with its functionalities and the code within itself and 3. Function Call, where the arguments are passed to the function definition by the caller of the function and if the function returns any value, then that value is assigned to a local variable where the function is being called from. The three integral parts of a function are return type, the identifier of the function and its parameters.

In this milestone values were passed as pass by value where the value is copied into the function parameter also known as function variables when the respective function is called along with their arguments. Pass by value functions receive a copy of the arguments passed. If the values of these parameters are updated within the function, they will not update the original values since the values were simply copied into the functions variables, which means a new memory block was assigned to these variables. Also, I came to know the meaning and usage of keyword ‘void’ in a function definition. A function can return only one value throughout its lifetime if a function is made of up of pass by value definition. If a function returns some value, we simply write the statement ‘return’ followed by the value that we want to return. Also, we need to specify the return type at the time of function definition and within a function’s prototype if the function is to return any value. Usually the return types are primitive data types but can also be user defined data types.

An explanation of term function and the need for functions in any language.

A Function is a piece of code which is a part of the program that performs a specific task assigned to it. It can perform the task assigned to it as many times as possible. For example, the function named as, “getInt” in MS2 validates the user input every time it is called in the program. It makes sure that user enters the correct input and only then, the control goes back to the caller of the function. This function can be called as many times as possible and will perform its task as defined within the function.

No software or application has ever been made without using at least one function throughout its entirety. Functions are the backbone of a program and the higher the number of functions in a program, the better the chances of efficiency of that program. Functions help,

(1) reduce the repetition of a code. If a logic/code is repeated more than once, that can be easily replaced by creating a function, which not only helps with the redundancy but makes the program clear to read, concise and improves the understandability of the program and its functionality.

(2) update the functionality at one place and the change is reflected throughout the code wherever the function is being called.

(3) breakdown a bigger problem into several parts and makes the code easier to manage.

(4) easily track a large program when divided into functions and helps easily find error with minimal effort as opposed to writing the code without a function.

(5) no limit as to how many times a function can be called, from anywhere throughout the program and make use of the same functionality whenever required.

Explanation on why the helper functions were not included in contacts module.

C language is a procedural programming language which uses procedures or functions to solve the problem. The contactsHelpers.c is a utility function file that has a cluster of functions whereas contacts.c can be called as core logic of the program. This helps separate the core logic and its functionality from the utility functions. This way of coding helps achieve readability of the code; the code gets easier to debug & updating the code becomes easy as we see when we add more functions to contactHelpers.c in MS4.

This type of coding can also be classified as modular programming. Modular programming means a set of functions which are used to construct a more complex program. The contact helper functions provide solution to the problem by breaking down the problem into several parts, for example, the yes() function, which plays a vital role in accepting the user’s choice whether they want to input a specific type of data or not. These functions are addressing a concern, or a requirement listed in the requirement document that help develop a solution or a program which is more complex.

The main positive factors for this type of coding are:

If the code is separated into multiple functions, then, when we need to update a functionality, we do not have to change the whole code, rather we only need to change a specific function where the change is necessary. This means that it would be considerably easy to find out which code needs to be changed without manipulating rest of the program.

Code that a programmer does not have to change is less likely to break than the code that he/she must change. If the program is separated into functions, it helps with the breakage of the functions that are unrelated to the change. However, if he/she does not create functions to address their problems in a program, then the possibility of breaking another functionality is very likely to happen.