RMIT University
School of Engineering
EEET2248 – Electrical Engineering Analysis

Lectorial Progress Report

Lecturer: Dr. Katrina Neville

Student Name: Aiden Contini Student Number: s3780445

Submission Due Date: 26th April 2019

Abstract:

This section of the lectorial required us to include user defined function(s) in our program, as well as ensure our program does not crash, and can handle erroneous inputs effectively. In this milestone, I chose to handle erroneous inputs via use of a flag. This allowed my program to be able to continually loop while the user input is invalid, as well as being able to print an error prompt to inform users of their mistake. One such erroneous input my program can now handle well include the invalid input of sub-zero numbers into any conversion other than the temperature. If the input field is left blank and the enter key is hit, code has been written to ensure the program does not crash for blank input, but instead the user is prompted to enter their input repeatedly until a valid input is entered. This error checking is all done through use of a while loop function, paired with a flag, to allow the loop to continue iteration while the input in invalid/empty.

In this milestone, initiative was also taken to change all variable names from the previous milestone. This change saw the "input" variable names standardised to be called the variable "in". This allowed for code to be reused throughout the whole program without the need for variable name changes. Standardisation could be done as many of the variables within the program are local variables, meaning they are independent of their values within other functions of the program.

Finally, this milestone saw the inclusion of separation through use of a user defined function. In this way, the code which runs the mathematic formula for the conversions is kept separately from the code which gets the user input and prints the solutions. In this way, standard coding techniques of separation are achieved, which sees code sectioned off and only drawn upon as needed, allowing different parts of the code to only actively undertake specific functions.

Software solution and testing :

As the formulas and formatting from earlier milestones are already proven to be working, solution testing for this milestone mainly involved ensuring the program does not crash regardless of what is inputted. The following table shows various outputs by the program showing how it handles erroneous inputs.

Explanation	Program Output
Inputted selection "19" from initial menu with only 18 items.	`Invalid input, please select a number from the list` `Would you like to restart the program? (y/n):`
Entered a series of blank inputs instead of entering a number to be converted. (User prompt is printed every time the input is left empty)	`Enter ºCelsius value to convert to ºFahrenheit:` `Enter ºCelsius value to convert to ºFahrenheit:` `Enter ºCelsius value to convert to ºFahrenheit:` `Enter ºCelsius value to convert to ºFahrenheit:` `Enter ºCelsius value to convert to ºFahrenheit:` `Enter ºCelsius value to convert to ºFahrenheit:`

Explanation	Program Output

Entered a negative number for a unit which does not support negative numbers (in this case centimetres)	`Enter centimeter value to convert to inches: -18`
	`Please enter a number greater than 0`
	`Enter centimeter value to convert to inches:`

When prompted for restart, a string other than ‘y’ or ’n’ was entered	`Would you like to restart the program? (y/n): hello`
	`Please enter "y" or "n"`
	`Would you like to restart the program? (y/n):`

The table has clearly shown the program’s handling of the various erroneous input that is possible at each step of the conversion, and hence is shown to successfully handle these without crashing. Unfortunately, due to Matlab’s general functionality, it is not possible to handle the error of a string inputted where an integer is expected. In this sole case, the default Matlab error is used, which prints a red error message and returns function to the keyboard (acting as a loop). Up until this point, the program would crash when incorrect inputs are entered, however now the program is able to loop and continually prompt for user input.

The overall usability of the program is very high, providing helpful prompts to users showing where they went wrong, as well as looping over itself and allowing users to correct input errors instead of crashing.