# Lab 4 [Main Menu]

In this workshop, you will continue you development of a text based game "Code Quest!". This workshop focuses on the use of functions to modularize and organize your code.

## LEARNING OUTCOMES

Upon successful completion of this workshop, you will be able to

- decompose a problem into two or more modules
- code a C function for each module
- implement structured programming principles, including single-entry/single-exit logic

## PART 1:

## SPECIFICATIONS

Write a program that contains a series of utilities for Code Quest! The program requires the following four functions:

- `void clearScreen()` : This is the clear screen function. This simple function prints out 40 newlines to clear the screen. It requires no parameters and returns nothing.
- `int validate(int low, int high)`: This is the validate function. This function prompts the user to input an integer. This function verifies that the integer is within a specified range (low and high), if not, the function displays a warning and prompts the user again. This function requires 2 parameters (low and high range) and returns the validated input from the user. You can assume the user will only enter numbers.
- `void newGame()`: This is the new game function. This is a placeholder function for a later lab. This function displays the massage "Not Implemented!". The functions requires no parameters and return nothing.
- `void load()`: This is the load function. This is a placeholder function for a later lab. This function displays the massage "Not Implemented!". The functions requires no parameters and return nothing.

Once the above four functions are implemented, write a program to

- Call `clearScreen` function to clear the screen.
- Display a menu to the user. The menu will allow the user to select either "1 - New Game", "2 - Load Game", "3 - Exit".
- Call the `validate` function to prompt the user for input, and validate the user's input. Note that the user's input must be within the range 1 and 3.
- Selecting New Game or Load Game calls their associated functions.
- The program only exits when the user selects Exit on the menu screen (looping required).

The output of a typical run-through of your program should look like this (user input highlighted in green).

```
--Main Menu--

1 - New Game
2 - Load Game
3 - Exit

Select: 6
Invalid input, try again: 0
Invalid input, try again: 1
Not Implemented!

--Main Menu--

1 - New Game
2 - Load Game
3 - Exit

Select: 5
Invalid input, try again: 2
Not Implemented!

--Main Menu--

1 - New Game
2 - Load Game
3 - Exit
Select: 3
Good Bye!
```

If your program's output exactly matches the output shown above, given the provided inputs, then your lab is complete and ready to be submitted (read below).

## PART 2 (BONUS)

## BUILDING THE GAME (OPTIONAL)

If you have completed all previous game labs, you may complete this section in order to merge your current lab with you previous labs.

- Copy the functions (and prototypes) into your codeQuest source code file.
- Copy the main code from this lab and paste it directly below code quest intro banner (before character creation).
- Move the character creation and battle scene code from the main and place it in the new game function, replacing the "not implemented" print statement.
- Clear the screen before the character creation section begins.

## Prepare a Typescript

Create a typescript on the remote Linux host using the following commands:

```
+ At the prompt, type: script w4.txt
+ At the prompt, type: whoami
+ At the prompt, type: cat w4.c
+ At the prompt, type: gcc w4.c –o w4.out
+ At the prompt, type: w4.out
+ At the prompt type: exit
```

This will produce a typescript named **"w4.txt"** in your current directory. Transfer a copy of this file to your local computer.

## SUBMISSION

Upload your typescript file to BlackBoard:

- Login to BlackBoard
- Select your course code
- Select Workshop 4 under Workshops
- Upload **`w4.txt`**
- Write a short note to your instructor
  - Under “Add comments”, add a sentence or two regarding what you think you learned in this workshop in the notes textbox
  - press "Save Changes"
- When ready to submit, press "Submit". Note you can save a draft until you are ready to submit.