CS 161 Example Design Document with Functions

Given the following hypothetical problem statement:

Problem Statement: Write a program that reads in up to 10 exam scores from the user and computes the average exam score. The exam scores may range from 0 to 100, and your program needs to check that the scores supplied are valid numbers before moving forward. This may include making sure the user doesn't enter a letter or string of letters.

- Ask the user for the number of exam scores they want to enter.
- Repeatedly ask the user to enter exam scores.
- After receiving the desired number of exam scores, output the average score.
- Use functions to break your program into smaller modules.

Here is an example Design Document:

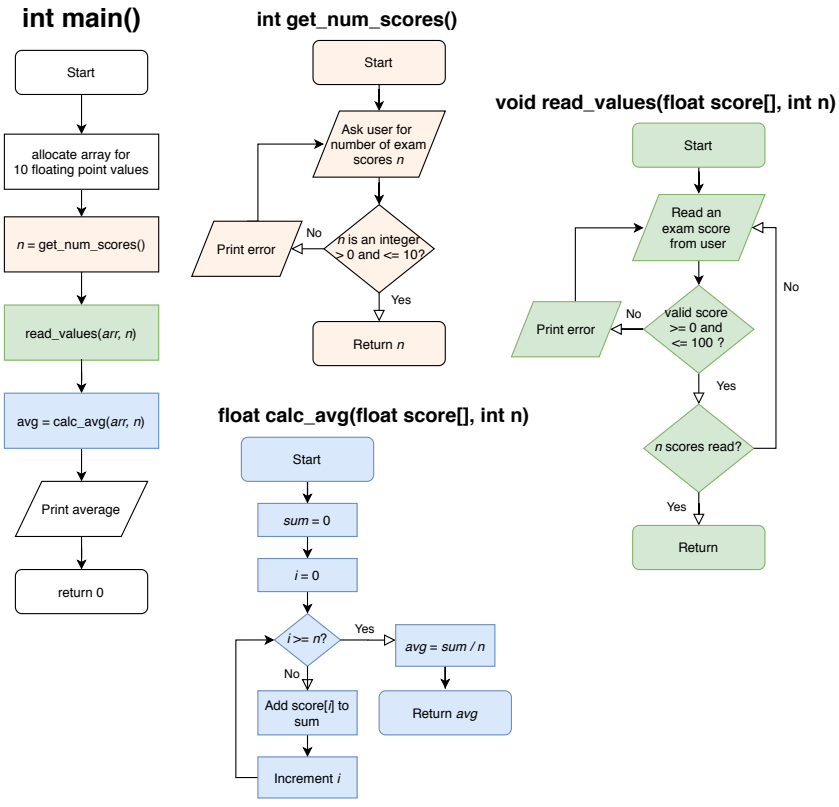
---

1. Understanding the Problem: My goal is to write a program that first reads an unsigned whole number value, *n*, from the user (must be between 1 and 10), and then reads *n* unsigned real numbers, which represent exam scores, from the user. These scores need to be between 0 and 100. If the user doesn't enter a valid number or a number in the correct range, then an error message is printed, and the user is prompted to enter a new number. After the user enters *n* valid real numbers in the range 0-100, then the average is calculated and printed to the screen.
Assumptions:

- I assume the user's number is an unsigned whole number.
- I assume the exam scores can be unsigned real numbers, not just integers.
- I assume that erroneous user inputs do not count against the *n* scores to be entered.
- I assume that the user will not need to change a previously entered exam score.

2. **Devising a Plan:** Here is my flowchart (at right).

Strategy: I predict that it will take me two hours to implement this program. I will start by implementing the get_num_scores() function to read in the number of exam scores from the user. Once that is working, I will add a check that it is a valid input, with an error and re-prompt if the input is invalid. Then I will add code in main() to allocate the array of 10 values and implement the read_values() function with a for loop to read in multiple scores. I will print out each one as it is read in as an aid for debugging (to be removed after the



program is fully tested). Finally, I will implement the calc_avg() function to compute the sum with a for loop and then compute the average and print it out.

3. **Test Cases:**

Setting	**Input**	**Expected Result**
Ask user for number of exam scores	-1	Print error and ask again
Prompt user for exam score	98.3	Store 98.3 in the array
User specifies 3 scores and enters them	98, 90, 92	Print average (93.33) and end
User specifies 3 scores and enters them. The second score has a typo and has to be re-entered.	98, 190, 90, 92	Re-prompt user after "190". When finished, print average of valid scores (93.33) and end.

(The more test cases you include, the better! Each one should test a different kind of setting+input combination to be equally useful.)