CS 161 Example Design Document

Given the following hypothetical problem statement:

Problem Statement: Write a program that reads in a series of 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, calculate the average and output it to the screen.

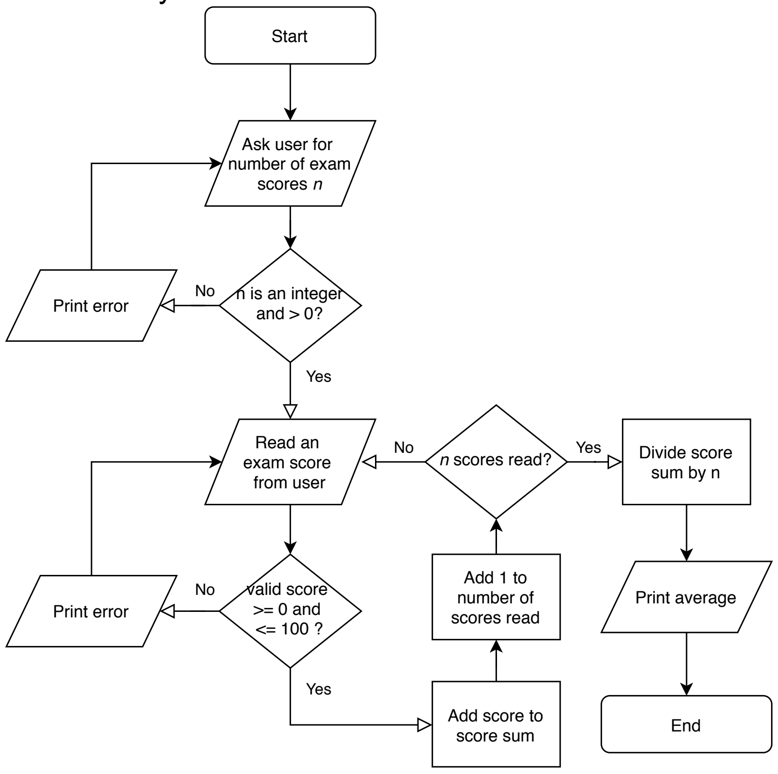
Here is an example Design Document:

---

1. Understanding the Problem: My goal is to write a program that first reads an unsigned whole number value, *n*, from the user, 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 number of exams 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 go back and change a previously entered exam score.

2. **Devising a Plan:** Here is my flowchart:



Strategy: I predict that it will take me two hours to implement this program. I will start by writing code 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 implement the 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 add code to keep track of the running total (and print it out as each score is read in, again for debugging purposes), then compute the average after all scores are entered 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	Add 98.3 to score sum and add 1 to number of scores read
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" and do not add it to the sum or the count of scores. 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.)