Isaac Stallcup
CS 161 Assignment #1

(**10 pts) Design for Assignment #2**

Design a solution for the following problem statement in Assignment #2. In addition, you will also provide a test plan for how you will make sure your program is working correctly. There is design in both the solution and testing your solution.

Problem Statement: Expand the problem from Assignment #1 to continue to ask the user for n test scores, rather than a set number of 5 test scores. What are the different ways you can think of doing this? Which design are you picking and why?

In addition, these test scores should 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. How are you going to make sure the user entered a number in the right range? What are the various ways you can think of making sure the user enters a number, rather than letters?

n scores:

Design one:

-After displaying test score averages header, declare variable `int testnum` of which the input is desired

-ask user for number of tests inputting, then get that number and set it as `testnum`

-in for loop, instead of `for(int iii = 0; iii < 5; iii++)` replace 5 with `testnum`

-when dividing fscoresum by five, again instead substitute `testnum` for five.

Design two:

-in for loop, set upper limit for `iii` to 64 or another arbitrarily large number

-when accepting inputs, add functionality to check whether input is a certain escape sequence then if the escape sequence is found, end the loop (for example if instead of a test score the user inputs `done` then the loop breaks)

-have outside variable tick up every time a test score is read, and then divide the sum of all test scores by that number

I am picking the first design, as it seems easier, simpler to code and less fallible. If the user in the second design wishes to stop the input, they have to input some letter string which may interfere with the program's new, requested functionality to accept only numbers. It is also more flexible, and its only limitation is having the user know beforehand how many test scores they have to input.

Error checking:

When each test score is input, check first if it is a numeric value, then if it is between zero and one hundred; this can be accomplished by the use of `if` statements, and possibly using a loop to increment through the input and check whether each individual character is a letter or a number

Test plan:

To test this plan, I will perform a similar series of tests as I did with the first program; inputting values that are not numbers or numbers <0 and >100 to test the portion of code telling the program to not accept input that isn't a number between 0 and 100.