Tommy Hollenberg
CS161 Assign3
Design Document
01/26/2017

Understanding the Problem

This C++ program will function as a grade calculator. The program must contain at least 3 functions, loops, and decision statements. The calculator must be able to calculate grades in specific categories (labs, assignments, tests, and recitations), as well as an overall grade for the class, given weights for each category. Lab, assignment, and test averages are all calculated by summing assignment point values and scores received, and dividing the later by the former. Within the recitation category, subcategory (quizzes, designs, and critiques) averages must be determined and weights must be given in order to calculate the combined recitation grade. If a weight is zero, then it can be assumed that the user doesn't want to enter any grades for that subcategory. The point value of individual labs/assignments/tests/quizzes/designs/critiques may vary. If so, then a point value must accompany each score input. If point values are uniform, then one point value can be input for all scores within the (sub)category. Grades, scores, and point values should all be floating point numbers. All category averages must be initialized to zero. The calculator should continue running until the user chooses to quit.

Devising a Plan

Lab, assignment, test, quiz, design, and critique average can all be calculated with the same function. The function should accept, as an argument, a reference to a double that holds the average for that (sub)category. Doing this allows asking the user whether they would like to keep the previously calculated (sub)category grade or overwrite it with a new calculation. The function should also have three (sub)category name string parameters: upper-case singular, lower-case singular, and lower-case plural. These strings will be used to customize the console output for each (sub)category. Finally, the function should optionally take a bool-type argument indicating whether or not this calculation is for a category or a subcategory (within the recitation average calculation). The default value of this parameter will be false. It will be assumed that if the user wants to recalculate the recitation grade, they will want to enter new data for quizzes, designs, and critiques, a true value passed to this bool parameter will cause the function to skip asking the user whether or not they would like to recalculate the subcategory average, but rather go ahead assuming that they do. The calculated average will be returned through the double average parameter passed by reference, so its return type will be void.

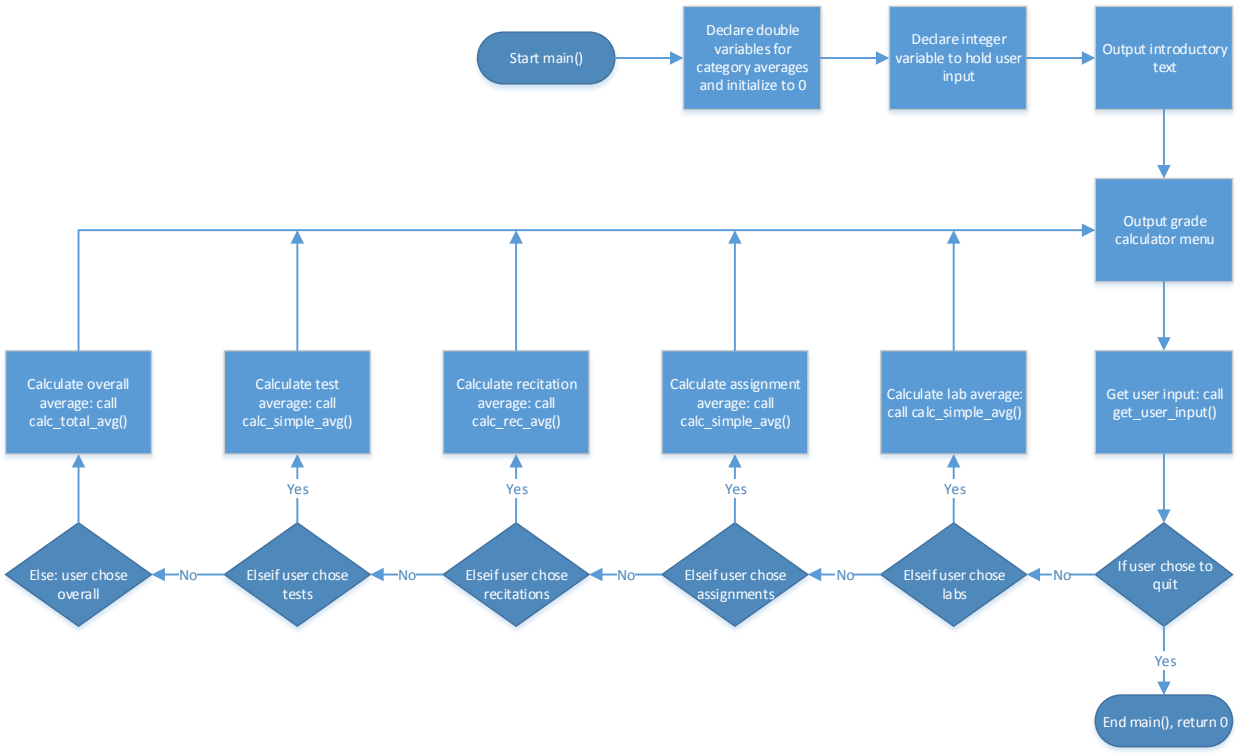
Example declaration:

```
void calc_simple_avg(double &average, string u_sing,
  string l_sing, string l_plur, bool subcat_flag = false);
```

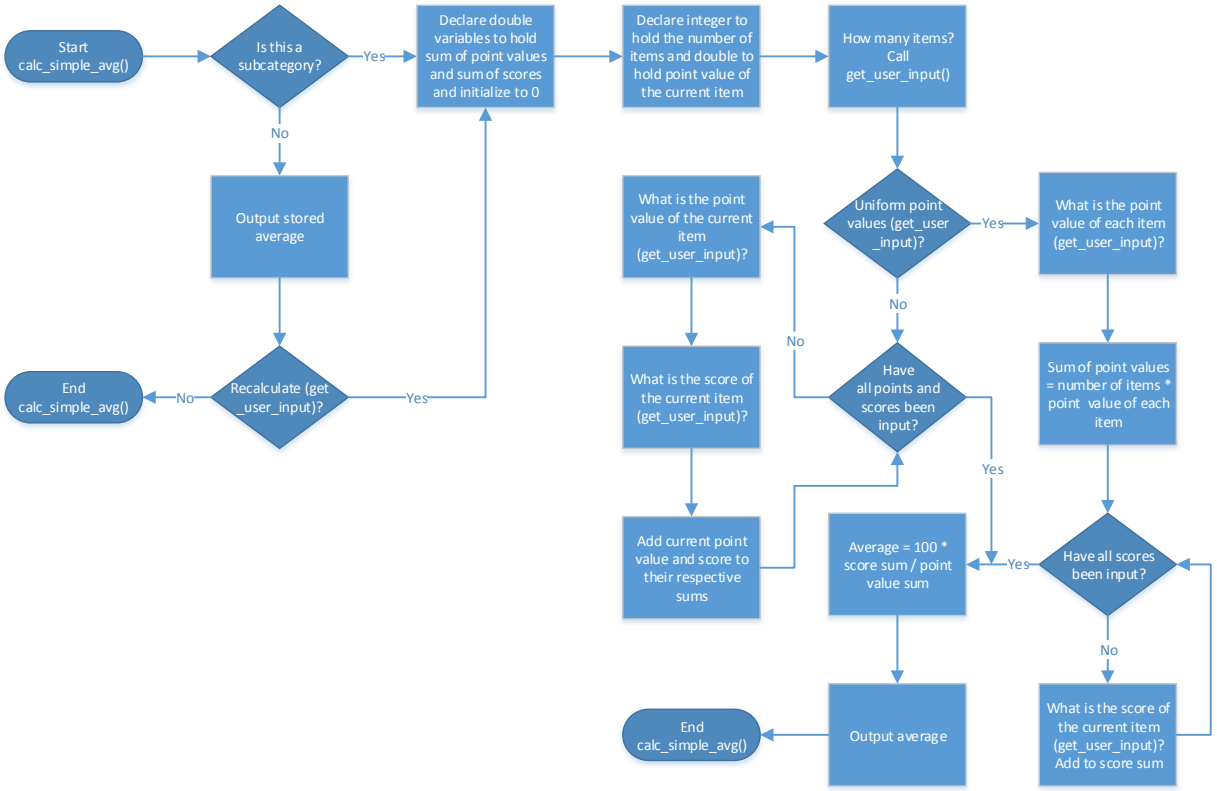
The functions to calculate the overall and recitation averages will both call the calc_simple_avg function in order to calculate the subcategory averages prior to combining them in a weighted average. Subcategory weights must sum to 100, and the functions will loop until this condition is met. If a weight is given to be zero, calc_simple_avg will not be called for that subcategory.

A function called get_user_input will be used for getting all required user input. It will accept, as arguments, a string holding a prompt to be repeatedly output until good input is received and, optionally, a double holding the maximum acceptable input value and a bool indicating whether only integer input should be accepted. The default value for the maximum input value will be the DBL_MAX macro from the cfloat library. The function will accept double values from 0 to the max input double, inclusively. If the integer-only bool is true, the function will round up to the nearest integer using the ceil function from the cmath library. If the rounded value is equal to the pre-rounded value, then the value is an integer and will be returned. When using this function with integers, INT_MAX should be passed as the max input value if no other maximum value exists to prevent overflow.

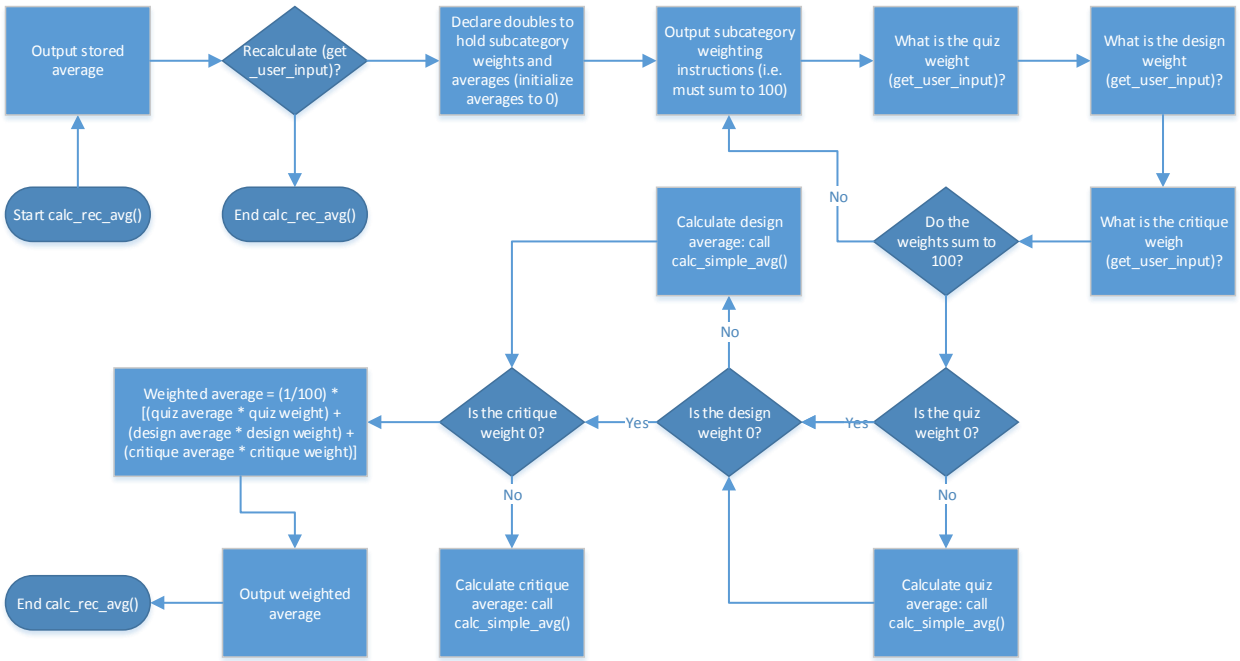
Main function:



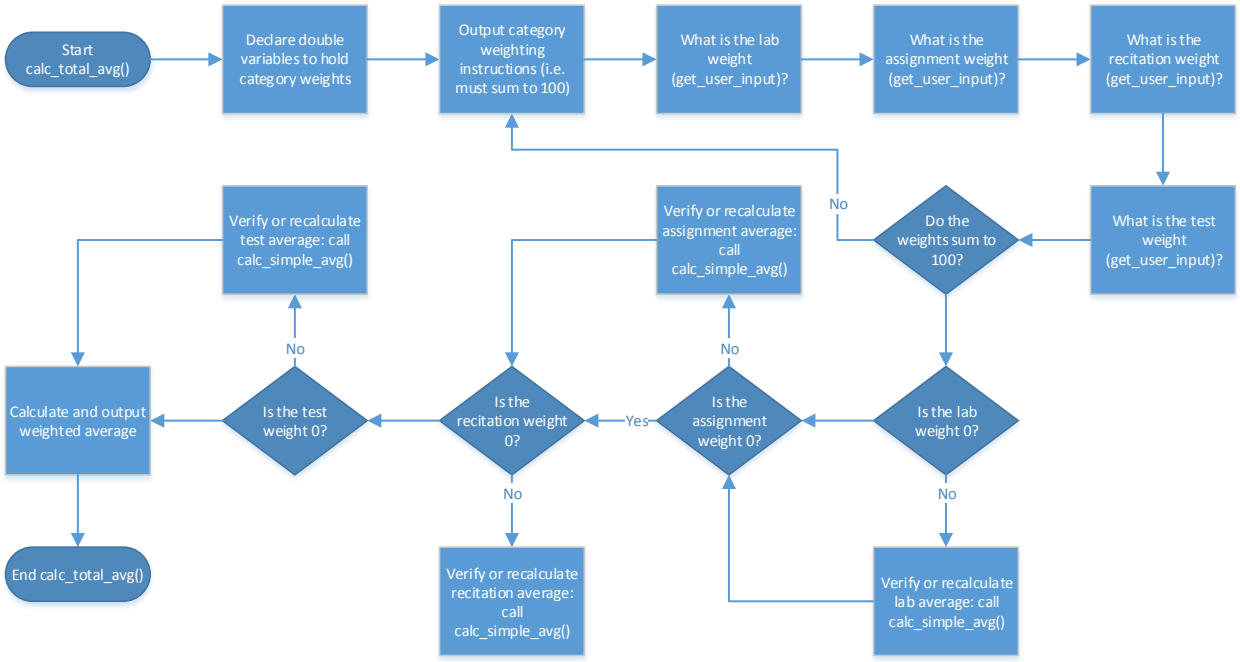
calc_simple_avg function:



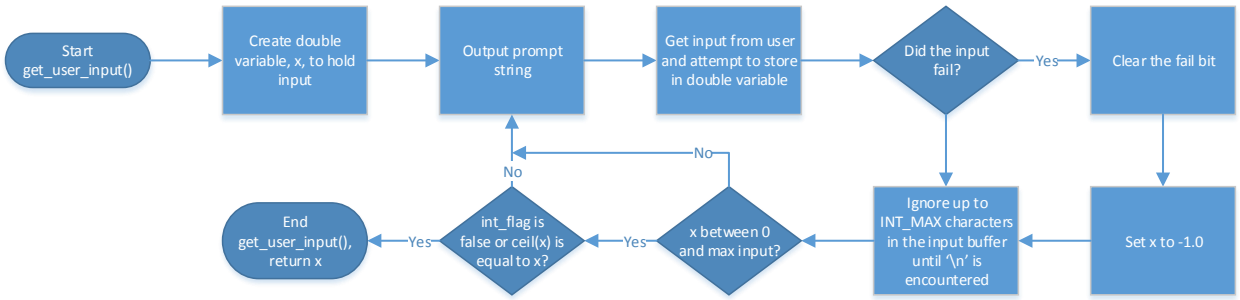
calc_rec_avg function:



calc_total_avg function:



get_user_input function:



Testing Plan

Testing get_user_input function: (assuming max input value is 5.0)

Input Value:	Expected Outcome:
1	Returns 1.0 to caller
1 (int_flag set to true)	Returns 1.0 to caller
2.6	Returns 2.6 to caller
2.6 (int_flag set to true)	Outputs prompt string again and waits for further input
1 2 1	Returns 1.0 to caller and ignores the rest of the input
1asd	Returns 1.0 to caller and ignores the rest of the input
22	Outputs prompt string again and waits for further input
/	Outputs prompt string again and waits for further input
asdf	Outputs prompt string again and waits for further input

Testing program logic:

Situation:	Input Value(s):	Expected Outcome:
Main menu	0	Quit
	1	Calculate lab average
	2	Calculate assignment average
	3	Calculate recitation average
	4	Calculate test average
	5	Calculate overall average
Simple average calculation:		
Recalculate average?	0	Continue with calculation
	1	Return to caller
How many items?	5	Ask for information about 5 items
	0	Ask for no information and output average of 0.0%
Uniform point values?	0	Ask for the point value of each item
	1	Ask for one point value for all items
Score of current item? (if point value is 10)	9	Move on to the next item
	11	Ask for the score again for the same item
Recitation average calculation:		
Recalculate average?	0	Continue with calculation
	1	Return to caller
Entering subcategory weights	40, 30, 30	Continue on to average calculations
	40, 10, 10	Ask for weights for all subcategories again
	40, 70	Ask for second subcategory weight again until the partial sum does not exceed 100, then asks for the third subcategory weight
	0, 30, 70	Skips calculation of the first subcategory
Overall average calculation:		
Entering category weights	10, 30, 20, 40	Continue on to average calculations
	10, 10, 20, 10	Ask for weights for all categories again
	40, 70	Ask for second subcategory weight again until the partial sum does not exceed 100, then asks for the third and fourth category weights
	10, 20, 0, 70	Skips calculation of the third category