Program #4

Due Nov 4 by 11:59pm **Points** 100

Designing Your Own Calculator

Introduction

For this homework project you will have two weeks to complete the programming portion of the assignment. **However, you will be required to submit a design document to Canvas (due on Sunday, October 28th, 11:59pm).** You can use the same documentation template and format that was introduced during week 2. Please review the recitation syllabus (https://web.engr.oregonstate.edu/~goinsj/resources/CS161/CS161_Recitation_Syllabus.pdf) (available from the Recitations tab) for additional details on the design document expectations.

Example design document (introduced in week 2): Polya_template.pdf (https://web.engr.oregonstate.edu/~goinsj/resources/CS161/Polya_template.pdf)

Notice that the design of this program is more open-ended than previous assignments. Oftentimes real-world assignments require the software designer to start from scratch and create a solution. For this assignment you must design a program that can solve the problem statement and you must then implement your solution.

(85 points) Problem Statement

There are many calculations that computers can help us with. You are to make a calculator program for this assignment which will allow users to do **standard calculations (+, -, *, /)**, **binary to decimal conversion**, **decimal to binary conversion** and a **grade calculator.** The user should have the option to choose which form of calculation they want to do. At the end of each calculation, the user should be asked if they want to do another calculation or exit the program.

The standard calculator should be able to handle floats and ints. It should take the equation on one line as a string.

For an example of the standard calculation:

```
What is your equation? 5 * -7
Result: -35.
```

Clarification (10/23/18): Here is an example that illustrates the requirement mentioned below that "...there should not be competing precedence. Everything will be calculated left to right." In essence, you just have to process the operators as you encounter them. This makes the assignment easier, rather than having to implement the normal mathematical order of operations.

```
What is your equation? 6 + 4 / -2.5 - 5.25
Result: -9.25
```

Explanation:

6 + 4 = 10
10 / -2.5 = -4
-4 - 5.25 = -9.25

Additional Requirements

- Clarification (10/23/18): Your code should detect cases where the number or ordering of operands is incorrect. For example, the calculation "6 + 4.125 + 7 9" would display an error, as would "6 + / 6".
- There should be spaces between each operand and operator. Clarification (10/24/18): You can assume that the user will include these spaces (e.g. the TAs will not test the calculator portion of your code with input strings that don't contain spaces).
- There should not be competing precedence. Everything will be calculated left to right.
- The binary conversions will only handle unsigned, positive values.
- The grade calculator should prompt for the number of grades to be entered, collect those grades from the user and average them. The user should also be given an option to calculate a weighted average. For this option the program will need to take a weight from the user and multiply it by the average. If you are unfamiliar with weighted averages, see the introduction at: **https://www.wikihow.com/Calculate-Weighted-Average (https://www.wikihow.com/Calculate-Weighted-Average)**

Clarification (10/29/18): For the weighted average calculation, it is sufficient for your code to accept a set of grades and associated weights to be applied.

Example for that part of the program:

```
You chose to compute a weighted average.
How many entries need to be averaged?
4
Please enter grade #1:
88
Please enter weight #1:
.25
Please enter grade #2:
95.6
Please enter weight #2:
.30
Please enter grade #3:
79w
Oops, invalid input.
Please enter grade #3:
79
```

```
Please enter weight #3:
.15
Please enter grade #4:
91
Please enter weight #4:
.30
The weighted average is: 89.83
```

- Your program should handle all input errors such as a user putting in an invalid character or number. e.g. To get full credit you should detect empty strings and invalid input such as "cat" or "k2.53". Additional error checking is extra credit (see the helpful hint section).
- All function bodies must be 15 lines of code or less, including main(). Whitespace and lines with just curly braces do not count in the line count.
- No global variables or goto functions.

Helpful Hint (10/31/18)

If you have started to work on the calculator code, it quickly becomes apparent that it is extremely difficult to do a good job validating the user's input line.

For example, each of the following input lines is invalid:
a5.2 + 3
5.2 + 3 /
5.2 * + 3

Due to the difficulty of catching all possible errors, I will add up to 10 points of additional extra credit to submissions that can properly reject any faulty inputs in the user's input. Remember, your code still needs to be written with <= 15 lines of code inside each function.

Examples of erroneous input that you need to detect for full credit:

5.4 * 24 + / 25
5.4 * 24 +
27.2 / 6.5 85
b28 * 3.5
2.4 / cat
(empty string)

Examples of erroneous input patterns that you can detect for extra credit:
5.8w + 2.5
23.5 / -7.8-
4.42 - 3..9

I uploaded some example code that illustrates how to catch many of these errors.
I suggest opening this file in a text editor that has syntax highlighting to make it easier to read.

It's available online at: **https://web.engr.oregonstate.edu/~goinsj/resources/CS161/demos/calculator_parsing.cpp (https://web.engr.oregonstate.edu/~goinsj/resources/CS161/demos/calculator_parsing.cpp)**

You are completely welcome to incorporate my example C++ code into your program as you see fit.

(15 pts) Program Style/Comments

In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the **style guidelines for this class (https://web.engr.oregonstate.edu/~goinsj/resources/general/cpp_style_guideline.pdf)** , and try to follow them, i.e. don't align everything on the left or put everything on one line!

```
/******************************************************
** Program: calculator.cpp
** Author: Your Name
** Date: 10/25/2018
** Description:
** Input:
** Output:
******************************************************/
```

Don't forget that each of your functions needs to have a header. For example:

```
/******************************************************
** Function:
** Description:
** Parameters:
** Pre-Conditions:
** Post-Conditions:
******************************************************/
```

(10 pts) Extra Credit

Submit an updated, annotated version of your design. How did your design change between when you submitted for recitation and when you submit for your assignment? Make sure to complete your testing table.

(10 pts) Extra Credit (added 10/24/18)

If you are looking for an additional challenge, write your code so that it can decode and correctly interpret parenthesis to indicate the preferred order of operations

For example:

```
What is your equation? ( 6 + ( 4 / -2.5 ) ) - 5.25
Result: -0.85
```

Assignment Submission

Note that there are two deadlines for this assignment.

Your design document must be scanned and electronically submitted to Canvas (in the assignment tab) by **Sunday, October 28th, 11:59pm**. You may use one of the document scanners (available in KEC1130 and some other College of Engineering labs) to scan your paper design into a PDF document that can be submitted to Canvas.

Electronically submit your C++ source code to **TEACH** **(https://engineering.oregonstate.edu/teach)** by the assignment due date on **Sunday, November 4th, 11:59pm**. Your TEACH submission should contain the .cpp file with your source code. If you choose to do the extra credit, you may also include a PDF file with your annotated program design and a list of the changes made to your design.

Reminder

Every assignment in this course is graded by demoing your work for 10 minutes with a TA. You are required to **meet with a TA within two weeks of the due date** to demo. You can schedule a demo with a TA from the **TA Office Hours** tab in Canvas. The available times can be viewed in the far right column of the table (labeled "Grading Hours"). Click on one of the links to access the poll and insert your name in a specific time slot.

- **Demo Outside 2 Weeks**: Assignments that are not demo'd within the acceptable time period will be subject to a 50 point deduction.
- **Demo Late Assignments**: Late assignments must still be demoed within the two week demo period beginning from the assignment's original due date.
- **Missing a Demo**: If you miss your demo with a TA, you will receive a 10 point (one letter grade) deduction to that assignment for each demo missed.

Each program needs to be written according to the **style guidelines** **(https://web.engr.oregonstate.edu/~goinsj/resources/general/cpp_style_guideline.pdf)** for this class. Remember that an important part of computer programming is making sure that your work is easily understandable by other programmers.

Program #4 Rubric

Criteria	Ratings	Pts
Program Header At a minimum, header should contain author's name and a description of the program. (2 pts each) 1 point for submitting code that compiles on flip.		5.0 pts
Good indentation / Use of whitespace Is code easy for the TA to read? Conditional blocks of code should always be indented.		5.0 pts
Each function is documented Every function contains it's own initial block comment (or multiple lines of comments prior to the function definition) that provides the reader with an explanation of the function's purpose. -1 pt for each function that is missing a header		5.0 pts
User has the option to select standard calculator, binary to decimal, decimal to binary, and grade calculator (2 pts each)		8.0 pts
All function bodies are <15 lines (excluding lines with just curly braces and lines with only comments). -3 points for each function over the limit		12.0 pts
All parts of the program handle error checking for the following input: <blank line> "cat" "." -2 pts for each section of code that doesn't reject all of those inputs		8.0 pts
User's calculator detects invalid input for: 10 10 +		2.0 pts
User's calculator detects invalid input for: 10 + *		2.0 pts
User's calculator detects invalid input for: + 10 10		2.0 pts
Calculator correctly computes 10 for: 2.4 + 7.6		2.0 pts
Calculator correctly computes -5.2 for: 2.4 - 7.6		2.0 pts
Calculator correctly computes 19 for: 2.5 * 7.6		2.0 pts
Calculator correctly computes 8.25 for: 41.25 / 5		2.0 pts
Calculator correctly computes 39.5 for: 50 / 5 * 7 - 25 + -5.5		7.0 pts
Binary to decimal calculator correctly converts 101010 into 42		4.0 pts
Binary to decimal calculator correctly converts 1001001 into 73		4.0 pts
Student's code correctly converts decimal 73 to binary: 1001001 (extra leading 0's are okay)		4.0 pts
Student's code correctly converts decimal 61680 to binary: 1111000011110000 (extra leading 0's are okay)		4.0 pts
User can choose to run grade calculator mode (average & weighted average)		2.0 pts

Criteria	Ratings	Pts
Grade calculator code prompts for number of entries (and uses the value correctly)		4.0 pts
Student's code correctly calculates average of 3 numbers		3.0 pts
Student's code correctly computes weighted average of 3 inputs		4.0 pts
No global variables or goto functions anywhere (these points are all-or-nothing)		4.0 pts
User can choose to repeat the program after each calculation		3.0 pts
Extra credit (10 pts): The calculator rejects the following input (2.5 pts each): "12 + 34w" "2.3. * 4" "5 * -7.7-" "3 / -23.j"		0.0 pts
Extra credit (10 pts): The standard calculator can process parenthesis to indicate the preferred order of operations. 4.5 - (9.0 / 3) should display 1.5		0.0 pts
Note that the recitation TAs will grade the 10 pts of extra credit for submitting an updated, annotated version of your design documents. This will not be checked during the demo.		0.0 pts
Total Points: 100.0		