

CptS 121 - Program Design and Development



Programming Assignment 1: Equation Evaluator

Assigned: Wednesday, January 17, 2018

Due: Friday, January 26, 2018 by midnight

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Analyze a basic set of requirements for a problem and derive logical solutions to them
- Declare variables
- Apply C data types and associated mathematical operators
- Comment a program according to class standards
- Logically order sequential C statements to solve small problems
- Compose a small C language program
- Compile a C program using Microsoft Visual Studio 2015
- Execute a program
- Create basic test cases for a program

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- 🐾 Access Microsoft Visual Studio 2015 Integrated Development Environment (IDE)
- 🐾 Summarize topics from Hanly & Koffman Chapters 1 - 2 including:
 - The steps of the software development method
 - C language elements (preprocessor directives, reserved words, and standard identifiers)
 - The standard C data types
 - The general form of a high-level program

III. Overview & Requirements:

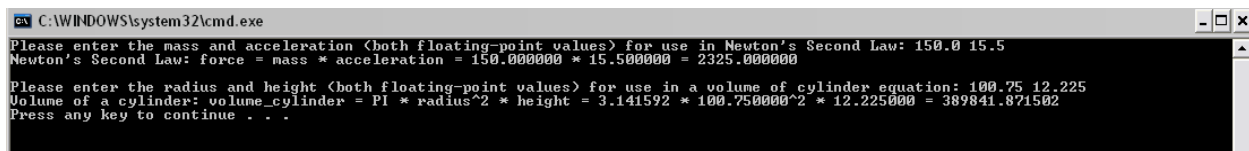
Write a C program that evaluates the equations provided below. All equations should be placed in a single project! The program must prompt the user for inputs to the equations and evaluate them based on the inputs. All variables on the right hand sides of the equations must be inputted by the user. All variables, except for the *plaintext_character*, *encoded_character*, and variable *a* are floating-point values. The *plaintext_character* and *encoded_character* variables are characters, and the *a* variable is an integer. PI, G must be defined as a constant macro (`#defined` constants). Error checking is not required for your program. You do not need to check for faulty user input or dividing by zero.

1. Newton's Second Law of Motion: $\text{force} = \text{mass} * \text{acceleration}$
2. Volume of a cylinder: $\text{volume_cylinder} = \text{PI} * \text{radius}^2 * \text{height}$
3. Character encoding: $\text{encoded_character} = (\text{plaintext_character} - 'a') + 'A'$ (note: what happens if *plaintext_character* is lowercase?)
4. Gravity: $\text{force} = G * \text{mass1} * \text{mass2} / \text{distance}^2$, where G is the gravitational constant with value $6.67 * 10^{-11}$
5. Resistive divider: $\text{vout} = \text{r2} / (\text{r1} + \text{r2}) * \text{vin}$
6. Distance between two points: $\text{distance} = \text{square root of } ((x_1 - x_2)^2 + (y_1 - y_2)^2)$ (note: you will need to use `sqrt ()` out of `<math.h>`)

7. General equation: $y = (89 / 27) - z * x + a / (a \% 2)$ (recall: a is an integer; the 89 and 27 constants in the equation should be left as integers initially, but explicitly type-casted as floating-point values)

IV. Expected Results:

The following console window illustrates inputs and outputs that are appropriate for your program. Your program must display the results in a similar form as shown in the window. The window shows possible results, for the given input tests, for the first two equations only.



```
C:\WINDOWS\system32\cmd.exe
Please enter the mass and acceleration (both floating-point values) for use in Newton's Second Law: 150.0 15.5
Newton's Second Law: force = mass * acceleration = 150.000000 * 15.500000 = 2325.000000
Please enter the radius and height (both floating-point values) for use in a volume of cylinder equation: 100.75 12.225
Volume of a cylinder: volume_cylinder = PI * radius^2 * height = 3.141592 * 100.750000^2 * 12.225000 = 389841.871502
Press any key to continue . . .
```

Note: you will need to display the results for all of the equations!

V. Submitting Assignments:

1. Using the OSBLE+ MS VS plugin, please submit your solution. Please visit <https://github.com/WSU-HELPLAB/OSBLE/wiki/Submitting-an-Assignment> for more information about submitting using OSBLE+.
2. Your project should contain your C source file (which must be a .c file).
3. Your project must build properly. The most points an assignment can receive if it does not build properly is 65 out of 100.

VI. Grading Guidelines:

This assignment is worth 100 points. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- 5 pts for correct declaration of constant macros
- 35 pts for proper prompts and handling of input (5 pts/equation)
- 49 pts for correct calculation of results based on given inputs (7 pts/equation)
- 11 pts for adherence to proper programming style established for the class and comments