COMSC-200 Lab 1
Problem Solving With C++

GOOD PROGRAMMING PRACTICES *show / hide*

ABOUT THIS ASSIGNMENT
In this lab session, you will write a "console program" that includes solutions to several geometry problems. Since we have not yet learned object oriented programming, you may use any C++ programming technique to solve the problems -- you do *not* need to use OOP; you do not even need to use functions. You just have to solve the problems using C++.

The purpose of this lab is 2-fold. First, it is to familiarize students with the non-programming, logistical details of editing, compiling, running, and submitting programs for COMSC-200. The second is to provide a basis for comparing and contrasting the OOP solutions which we will develop over this semester, with non-OOP solutions.

Click here to see reference for compiling in C++. These explain in detail the steps in creating a program, using a variety of systems and compilers. These steps apply to all of the lab work for COMSC-200. Future lab sheets will *not* contain the this link.

Labs 1-10 include assignments from out of our Deitel textbook. There are multiple editions of the book, and just about all of them are suitable for use in this course. But the reading and lab assignments are based on page, chapter, and section reference number from the 7th, late objects edition. So to make it easier for students, each lab writeup (including this one) contains a link to a PDF with scanned pages of the referenced material from the book. Use the links that refer to "figures", like the "figures 3.7 and 3.8" link in the lab 1a writeup below. The same materials are publicly available as CPP and H files at media.pearsoncmg.com.

As you complete this assignment, post the required files to the COMSC server. You may post them all at once, or one-at-a-time as you complete them, as you prefer. The individual labs are graded in order, starting with "a", which must be fully correct before "b" is graded, and so on. This assignment is worth 50-points, to be awarded after all labs are successfully completed. Use the "Submit your SP2015 work" link on the class website to post your file for this lab, using the **lab1** folder provided.

Submit only the specified file. Do *not* submit executables or input files, except as specified in an assignment.

Lab 1a [`MyGrade.txt and MyGrade.cpp`]
Write the **pseudo code** and the **CPP** program as presented in figures 3.7 and 3.8 of our Deitel textbook, with modifications per the requirements listed here. Do *not* save the pseudo code in a DOC, DOCX, or other rich text format -- use a simple text editor. Name the file with the pseudo code **MyGrade.txt**, and the CPP **MyGrade.cpp**. The purpose of this lab is to establish some "universal requirements" that will apply to all work that you do in this course, in addition to the requirement already established for identifying information.

UNIVERSAL REQUIREMENT #1: Use the C++ Standard header files, without the **.h** extensions. Do *not* use `using namespace std;`. Use statements like `using std::cout;` instead. This applies to *all* lab work in this class.

UNIVERSAL REQUIREMENT #2: Spell and case filenames *exactly* as specified.

GRADING POLICY: Remember to follow the grading policy checklists included at the end of each lab writeup!

*Post **MyGrade.txt** and **MyGrade.cpp** to the COMSC server for credit.*

Lab 1b [`GeometryHomework1.cpp`]
Here is the specification for the GeometryHomework program.

REQUIREMENTS: Write a C++ program to make area, perimiter, surface area, and volume calculations, in any order or combination. The calculations for 6 geometric shapes include: (reference: Geometric Formulae Review).

- area and perimeter of a square
- area and perimeter of a rectangle
- area and circumference of a circle
- surface area and volume of a cube
- surface area and volume of a rectangular prism
- surface area and volume of a cylinder

INPUT: The program input shall be from a text file named **geo.txt**, to be located in the same folder as the executable. The file shall consist of an unspecified number of lines. Each line of the file will specify one geometric object, or be blank. The line format consists of (1) an object type in uppercase (SQUARE, RECTANGLE, CIRCLE, CUBE, PRISM, or CYLINDER), and (2) the dimensions of the object, in the following order:

for square, 1 dimension: "side"
for rectangle, 2 dimensions: "length" and "width"
for circle, 1 dimension: "radius"
for cube, 1 dimension: "side"
for prism, 3 dimensions: "length", "width", and "height"
for cylinder, 2 dimensions: "radius" and "height"

LINE FORMAT: A line of input may consist of up to 4 (four) usable tokens, each separated by one or more spaces. Ignore any tokens past the fourth token, if they exist. The first token is the object name (for example, "SQUARE"). The second through fourth tokens are dimensions. Some calculations need only one dimension (CIRCLE, SQUARE, and CUBE); others need two (RECTANGLE and CYLINDER) or three (PRISM). There may be blank lines, which are to be ignored.

It is possible for an invalid object to be specified, or for the wrong number of dimensions to be supplied. Here is a sample input file (which includes a blank line, and one pair of dimensions separated by more than one space):

```
SQUARE 14.5
RECTANGLE 14.5    4.65
CIRCLE 14.5
CUBE 13
PRISM 1 2 3

SPHERES 2.4
```

```
CYLINDER 1.23
CYLINDER 50 1.23
TRIANGLE 1.2 3.2
```

RESULTS: Output the object name, its dimensions, and calculated results *on one line*. If the object is not recognized as one of the 6 allowable object types, produce output to this effect: `TRIANGLE invalid object`. If the object is recognized, but there are not enough dimensions provided, use the default value of *zero* (0) for each missing dimension. The results for the above input should be as follows. Note the identifications of the dimensions and the calculated values. The *dimensions* and *calculated results* may appear in any order:

```
SQUARE side=14.5 area=210.25 perimeter=58.00
RECTANGLE length=14.5 width=4.65 area=67.43 perimeter=38.30
CIRCLE radius=14.5 area=660.52 circumference=91.11
CUBE side=13 surface area=1014.00 volume=2197.00
PRISM length=1 width=2 height=3 surface area=22.00 volume=6.00
SPHERES invalid object
CYLINDER radius=1.23 height=0 surface area=9.51 volume=0.00
CYLINDER radius=50 height=1.23 surface area=16094.37 volume=9660.39
TRIANGLE invalid object
```

ACCURACY: Format the calculated results to 2 digits after the decimal place. Echo the input values *without formatting*. This code sample shows how to turn formatting on -- use it before outputting the *calculated results*:

```
cout.setf(ios::fixed|ios::showpoint);
cout << setprecision(2);
```

This code sample shows how to turn formatting off -- use it before outputting the *dimensions*:

```
cout.unsetf(ios::fixed|ios::showpoint);
cout << setprecision(6);
```

These require `using std::ios;` in the iostream library. It is compatible with Microsoft C++, Mac XCode, Cygnus g++, and ideone.com compilers.

Hints: Click here for sample code to parse a text file. Click here for sample code to convert text to numbers. Click here for sample code to format numbers sent to `cout`.

NULL POINTERS: Beware! The C++ specification says for the atof function *"If the parameter does not point to a valid C-string, or if the converted value would be out of the range of values representable by a double, it causes undefined behavior."* In this lab, it's possible for token[1] and the others to be NULL, if there is a missing dimension. So before doing atof(token[1]) or something like that, check to see if the parameter is NULL. If it is, use zero for the dimension and so NOT call the atof function.

INCLUDES: Beware! If you use items from the C and C++ libraries, use the proper #includes! If you don't know where `atof` comes from, google "atof library include" and find out. Do NOT assume that you've got it covered just because your CPP compiles for you. And for items from C libraries, like `atof`, do NOT put "using std::" statements -- that's only for things from C++ libraries that are defined in the "std" namespace.

Compile and run the program.

*Post **GeometryHomework1.cpp** to the COMSC server for credit.* Save this file, because modifications will be made to this program in labs 2-4 and 11-13. The GeometryHomework series will be the subject of seven (7) lab assignments.

How to pause a console program: *show / hide*

GRADING POLICY *show/hide*
