General Instructions

Programs should exhibit good programming style. Specifically, use consistent and appropriate indentation and blank lines (put blank lines where they improve readability but eliminate irrelevant blank lines).

I often use batch files to grade programs.

- The test bed uses Microsoft Visual Studio to compile your programs. If you use a different compiler to create your programs, please test them with Visual Studio before submitting them for grading.
- The automated test bed will fail if you do not follow the naming instructions, so please be sure that the programs are named correctly.
- The test bed automatically feeds input to your program when it executes. This means that your program must prompt for and read the input in the order specified in the assignment and that it must not perform prompts or reads that are not specified in the assignment. Do not execute a pause or “dummy” read at the end of the programs. Run the programs “Without Debugging” and the console window will not close prematurely. Do not put the code in a loop prompting to do an additional calculation or to correct erroneous input.
- Unless specified in the assignment, you may format the output as you wish.
- Please check to make sure that you upload the correct files. Initially, you will upload source code files that end with .cpp and later you will also upload header files that end with .h. Do not upload solution files that end with .sln (these do not contain the source code and cannot be graded).
- The first programming labs consist of multiple programs. You must upload all programs before pressing the “Submit” button. Once the submit button is pressed, you cannot upload additional files. However, you can “take back” an assignment before the due date and resubmit it.
- Do not zip the files together.
- All files are uploaded to Blackboard / WSUOnline (http://online.weber.edu/webct/entryPageIns.dowebct)
- When all files are uploaded, press the “Submit” button. I cannot see or access your files until you press the Submit button.

Program #1

Quadratic equations have the form: \( ax^2 + bx + c = 0 \) where \( a \), \( b \), and \( c \) are the coefficients and \( x \) is the unknown value. The two roots are computed using the quadratic formula:

\[
x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}
\]
The math library includes a function named `sqrt` to calculate square roots. This library function is described on page 65 and demonstrated in hypot.cpp.

There are two possible errors that you will ignore for now but will deal with in the next assignment. First, if ‘a’ is 0, the quadratic formula divides by 0 and fails. Second, if the roots are imaginary, the formula tries to take the square root of a negative number and fails. In both cases, you will see output that looks like -1.#IND or -1.#INF. If you use the test cases below, you will not encounter either problem.

Write a program named `roots.cpp` that:
1. prompts for and reads the coefficients in the order a, b, and c
2. calculates and prints two roots
3. do not include loops or any other prompt or input in your program

Test Cases:
- a = 2, b = -8, c = 6; x1 = 3, x2 = 1
- a = 3, b = 1, c = -6: x1 = 1.25733, x2 = -1.59067

Program #2

Write a program named `cone.cpp` that:
1. prompts for and reads (in this order) a radius and a height
2. calculates and prints the volume of a right cone: \( v = \frac{1}{3} \pi r^2 h \)
3. calculates and prints the total surface area of the cone (base + cone): \( S = \pi r^2 + \pi \sqrt{r^2 + h^2} \)

The math header file `<cmath>` defines a symbolic constant for \( \pi \): M_PI (which you must use). The Microsoft compiler requires a non-standard definition before math constants may be used:

```cpp
#define _USE_MATH_DEFINES // Microsoft only
#include <iostream>
#include <cmath>
using namespace std;
```

You may use the pow function (included in the math library and demonstrated in class) to square the values in the above formulas; however, for squares and cubes, it’s just about as easy to multiply the numbers by themselves: `radius * radius`.

Test Case:
- radius = 1.5, height = 3; volume = 7.06858, surface area = 22.8744