10 points.
Due: Friday, October 01, 2021at 11:59 pm

Section 1 Program Summary

For this first "real" program of the semester, you'll create a simple utility to help with geometry homework. Specifically, you'll write a program to calculate the areas of circles, triangles, and rectangles, as well as volumes of spheres, cones, and boxes.
Since we're sticking to command-line (GUI coding is its own entire topic), we'll try to create something reasonably user-friendly with a menu-based interface. Each menu will have a few options, and will wait for the user to enter an option. The program will then respond based on the selected option.

You'll also get some practice writing functions in C++. We will provide a header file with function headers for calculating each area/volume required for this program. You will need to write the function bodies, and set up the right include statements to call those functions in your program.

Section 2 Program Details

In this section, we'll give details on how to implement the program outlined above.

(a)

Menus

The menus and menu items for the program will be as follows:
1. In the main menu, the options are to solve a 2D or a 3D problem.
2. If the user chooses "2D," they will get a new menu asking to calculate the area of a circle, triangle, or rectangle.
3. If the user chooses "3D," they will get a new menu asking to calculate the volume of a sphere, (right circular) cone, or box.

The menus should each include an "exit" option as well. For the main menu, "exit" will end the program. In the sub-menus, an exit should return the user to the main menu.

When the user chooses a shape in the 2D or 3D menus, they should be prompted for the input measurement(s) needed to perform the calculation (e.g. a radius, if they chose to calculate the area of a circle). The program should then output the calculated value, and prompt the user to enter "any number". This ensures the user can read the result before the program outputs anything else. Once the user enters a number, they should be returned to the main menu.

(b)

Menu Inputs/Formatting

Each menu item must have a number indicating the value for the user to enter to pick that menu item (see example menus below).

If the user enters a number that does **not** correspond to a menu entry, your program should re-prompt the user and read in a new number. This "re-prompt" could either re-print the entire menu, or print a new prompt such as "Invalid choice, please input a valid menu item." You can choose which approach to take, so long as your program does not wait *silently* for valid input. Continue to prompt the user until the number they enter is a valid menu item.

Below, we've given examples of each menu, which show what items the menus should have, and in what order they should go. Your program is **not** required to print exact copies of the examples here (though you're welcome to copy the example formats if you like), but you **must** keep the menu items in the same order. While exact text of each menu item is up to you, the items should have reasonable labels. *For example*, your main menu's first entry could say "`1. 2D Menu`" instead of "`[1] Calculate a 2D Area`," but not something unrelated like "`Spider 2 Y Banana`" (no matter how much you like Jon Gruden).

Here's an example of the main menu:

```
Welcome to Geometry Helper!
[1] Calculate a 2D Area
[2] Calculate a 3D Volume
[3] Exit program
Please enter a menu item:
```

And an example of the "2D" menu:

```
Choose an Area to Compute:
[1] Circle
[2] Triangle
[3] Rectangle
[4] Return to main menu
Please enter a menu item:
```

The "3D" menu:

```
Choose a Volume to Compute:
[1] Sphere
[2] Cone
[3] Box
[4] Return to main menu
Please enter a menu item:
```

Finally, an example of a "completed calculation":

```
The area of the circle is: 3.14
Enter any number to return to menu.
```

(c)

Inputs for area/volume calculations

When a user selects a shape in the 2D/3D menu, your program should prompt for the shape's width, height, etc. depending on the shape.
As with the menus, you can choose the details of your prompt, and how many prompts to do (e.g. you could ask for a triangle's width and height in one prompt, or get width first and then prompt for height). Just make sure that your prompt is meaningful
For example, "`Please enter the radius of the sphere: `" would be a reasonable prompt;
"`Tickets, please: `" would not (no matter how much you liked Never Ricking Morty).
Once you've collected the necessary inputs, you can use them and call the appropriate function from `functions.h` to get your area/volume.

Note: for the Cone shape, you **must** read the radius first, and the height second. This minimizes the risk of us entering numbers in the wrong order during grading and getting incorrect outputs.

You can assume the user will give you the right type of input (i.e. they'll only enter numbers), but you must check that the values themselves are ok. Since we're dealing with physical measurements, negative values are not valid. When an invalid value is given, your program should re-prompt for a new input until you get a valid number. This is similar to what we said about menus and waiting for the user to pick a valid menu entry.

(d)

Functions

As mentioned previously, you will write functions to calculate each area/volume for the shapes listed above. When you start work on your program, you should download a copy of `functions.h` from Canvas. You need to create a `functions.cpp` file and implement the functions.
Then, ensure your code in `main()` calls the appropriate functions in `functions.h` to calculate the area/volume for each shape.

Do not change the function headers, as we need to be able to call the functions in our grading script.

(e)

Math Formulas

In case you've been out of math classes for a while, here are the equations you'll need:

- Circle: $A = \pi r^2$
- Triangle: $A = \frac{1}{2}wh$
- Rectangle: $A = wh$
- Sphere: $V = \frac{4}{3}\pi r^3$
- Cone: $V = \frac{1}{3}\pi r^2 h$
- Box: $V = whl$

You can approximate $\pi = 3.1416$.

Section 3 Submission

Submit a zip file, which contains two .cpp files called `main.cpp` and `functions.cpp`, as well a copy of the `functions.h` file.

Your zip file should be named *wiscID*`_asg1.zip`, where *wiscID* is your Wisc/Net ID (the username you use on Canvas, *not* the 10-digit number on your wiscard).

In each cpp file, you must include a comment at the top of the file with your name and Wisc ID.

When main.cpp is built and run, it should print out a menu similar to the examples, and allow the user to interact with the menus as described above.

Section 4 Hints

`enum`s and `switch`es are well-suited to handling menu inputs.

`do...while` loops are well-suited to waiting for valid inputs.

In lecture 3, we'll learn about writing functions in C++. Technically, this program releases before we learn functions (in the future, programs will only cover content from previous lectures). We recommend

you begin working on the code for navigating menus, and then start writing your functions after lecture 3.

While not required, you are welcome to add new functions to `functions.h` and `functions.cpp` as desired. Just don't change any of the functions that are already there.
For example, it can be very helpful to wrap the code for printing each menu into separate functions (such as `PrintMainMenu`, `Print2DMenu`, etc.).

Section 5 Rubric

Item	**Points**
Program builds and runs without compile errors	2
Program has a main menu, with all 3 options specified	1
Menu for 2D calculations has all 3 required shapes, in the correct order	1
Menu for 3D calculations has all 3 required shapes, in the correct order	1
Area calculation functions are implemented correctly	1.5
Volume calculation functions are implemented correctly	1.5
Program re-prompts for new input when users give invalid inputs	1
Program waits for user to enter a number after printing a calculation	0.5
Program gives reasonable prompts when asking for user inputs	0.5
Total	**10**