ROB 502 Fall 2022: Assignment 1

Due Friday 9/14 at 11:59pm

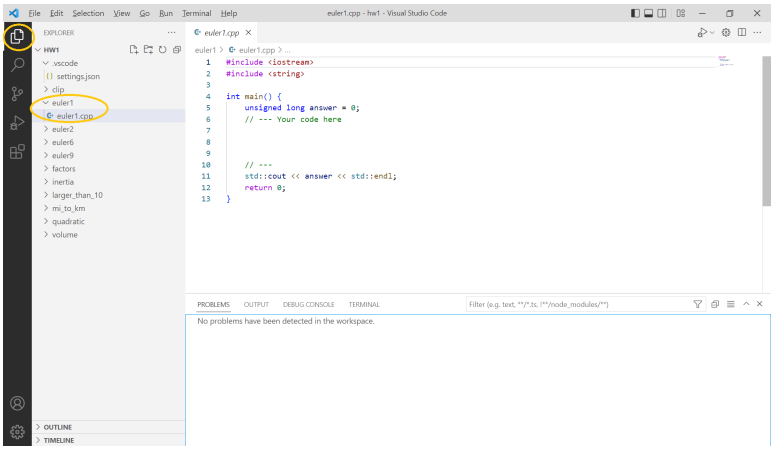
Rules:

1. All homework must be done individually, but you are encouraged to post questions on Piazza
2. No late homework will be accepted (unless you use your late-day tokens)
3. Submit your code on autograder.io
4. Remember that copying-and-pasting code from other sources is not allowed

Code Download:

Download the template code zip here. Unzip the the contents of the zip file to your Dropbox folder. Make sure you have set up VScode before starting the assignment. For instructions on VSCode setup, see this guide.

Open the folder `hw1` in VScode. You can do this by `cd`-ing to the `hw` folder and running `code .` . To test that your VScode is setup correctly, start by opening `euler1/euler1.cpp` from the Explorer panel on the left:



and then pushing F5.

When you write code for the problems below, make sure to open the `hw1` folder as above and then edit the code (the `hw1` folder contains some VScode settings that you won't get by opening one of the subfolders directly).

Instructions:

Each problem will give you a file with some template code, and you need to fill in the rest. We use the autograder, so if you're ever wondering "will I get full points for this?" just upload your code in the autograder to check. There is no limit to how many times you can upload to autograder. The autograder is set to ignore whitespace and blank lines, so there is some forgiveness on formatting. In most of these questions, the input/output and printing is handled for you in the template code, but in some you are asked to write it yourself. The autograder may test your problem with multiple different inputs to make sure it is correct. We will give you examples of some of these inputs, but the autograder will also try your code with some other inputs. The autograder will only show you if you got it right/wrong, so if you don't get full points, try to test some other inputs and make sure your program works.

Problems:

1. **Inertial Matrix Calculator** [2 points]: Start with the template code in `inertia/inertia.cpp`. Don't edit the code that's already there, and only write code between `// --- Your code here` and `// ---`. The templates will handle reading the input and printing the output, your task is just to write the logic of the program. Write a program that takes as input the mass and dimensions of a box, and outputs the inertial values `Ih,Iw,Id` according to https://en.wikipedia.org/wiki/List_of_moments_of_inertia (solid cuboid). The program will read the input values of mass, width, height, and depth from the file `inertia_input.txt` which is given to you. The template code handles the reading and printing, your job is to translate the equations for `Ih`, `Iw`, and `Id` into code. Template file: `inertia/inertia.cpp`
2. **Quadratic formula calculator** [8 points]: Write a quadratic formula calculator using `quadratic/quadratic.cpp`. The program will take three numbers from std::cin and print the solution(s) `x` to `std::cout`. If there are two real solutions, print the smaller

solution first, one number per line. If there is only one solution, print it only once. If there are no real solutions, print “None”.
Template file: `quadratic/quadratic.cpp`

Example inputs:

- `5 6 1`
- `4 7 2`

3. **Project Euler Puzzles**. For each problem, click on the link to see the description of the puzzle. Fill in the solutions in the corresponding files. There is no user input for these programs.
   a. Project Euler #1 [2 points], **Multiples of 3 or 5**.
   Template file: `euler1/euler1.cpp`
   b. Project Euler #2 [2 points], **Even Fibonacci numbers**.
   Template file: `euler2/uler2.cpp`
   c. Project Euler #6 [2 points], **Sum square difference**.
   Template file: `euler6/euler6.cpp`
   d. Project Euler #9 [2 points], **Special Pythagorean triplet**.
   Template file: `euler9/euler9.cpp`
4. **Control flow practice**. In this question you will practice translating simple algorithms into C++. The algorithms are meant to be simple with no gotchas. The autograder will run several inputs and check the output, so make sure your program works correctly with multiple different inputs.
   a. Algorithm [10 points]: Determine whether a number is a factor of 2 or 3.
   When you run the code, you will need to type in the number you want to test in the “TERMINAL” and press enter. Make sure you test a few different inputs!
   Template file: `factors/factors.cpp`

   Example Inputs: number from `1` to `5`
   b. Algorithm [2 points]: Compute the volume of a cone given radius and height.
   You will need to type in radius and press enter, then the height and press enter.
   Template file: `volume/volume.cpp`

   Example Inputs: `1 1`

c. Algorithm [2 points]: Check whether a value is within a given range (inclusive), and if the “clip” flag is set, and it’s outside the range, clip it.
This program will read input from a file called `clip_input.txt`, each line represents one “test case” in the format `min_value max_value value clip`. The first three are integers, and clip is a 0 or a 1 which will be read into a `bool` variable. The `clip_input.txt` file starts with only a few test cases in it, "0 10 15 0". In this case 15 is larger then 10, but clip is 0, so the correct output is 15. You should add more test cases to ensure you code is correct.

   Template file: `clip/clip.cpp`

   Example Inputs: `clip/clip_input.txt`

d. Algorithm [6 points]: Loop through an input file and print the first number larger than 10.
When you run the code, you will need to type in the filename you want to test. For instance, we provide `larger_than_10.in` as an example. The template code handles creating the `std::ifstream` object, which can be used to loop over the file. HINTS: try using a `while` loop, and the `.eof()` method.

   Template file: `larger_than_10.cpp`

   Example Input: `larger_than_10.in`

5. **Chapter 3, Exercise 2** [2 points]. “Write a program in C++ that converts from miles to kilometers”. This time the template file is completely blank, you have to write everything from scratch. Please print the number of kilometers to 3 decimal places (Hint: use `std::setprecision(3)`). The input is single number you should read from `std::cin` as in earlier problems, and you should use a `double` type to represent the input and output.
   Template file: `mi_to_km.cpp`

   Example Inputs:

   - `1`
   - `0.621371`

6. **Extra credit** [1 point]: Write a script that prints the the last time your dropbox synced. You can manually hard-code the time, but you have to use C++ date/time functionality (Hint: std::chrono). There’s no “right” answer, any submission that runs

without error will receive 1 point. This is just for fun, and to make sure you have your dropbox syncing correctly. Please name the file `dropbox.cpp`

If you don't do the extra credit, the autograder will warn that you're not uploading a file called `dropbox.cpp`. That's fine! You can still get full points.