ROB 502 FA24: Homework 1

See Canvas for due dates.

Rules:

1. All HW must be done individually, but you are encouraged to post questions on Piazza and discuss during OHs (recall: do not email assignment questions to teaching staff!)
2. Late work adheres to the late grading policy (see syllabus, use 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:

The code is available on the ROB 502 FA24 Git repo in the appropriate HW folder.

We prefer that you use the ‘`fetch`’/‘`pull`’ Git methods for the HWs, but we know that Git can be hard/confusing; if necessary, you can always download the code from the repo link. However, learning to use Git will be to your benefit for multiple reasons:

1. You can get some easy extra credit if you properly pull and version-control (and comment) your work for the HW assignments!
2. ROB 550 will require you to use Git in a much more complex capacity, and without much learning time (large code databases, lots of teamwork with multiple branches, etc.).
3. If you ever need to setup a new VM or use your code on another device (new computer, VM crashes, etc.) you can easily restore your code progress with Git.
4. Many robotics-related industries and research labs rely on Git for robust version-controlling of their work… you can put this on your resumé!

Open the directory `hw1` in VSCode. You can do this by `cd`-ing to the `hw1` directory and running `code .` . Do not run VSCode from subdirectories of `hw1` .

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 are 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.

Sample input and output are given as `input.txt` and `output.txt` respectively. Getting your program to successfully compile and correctly reproduce the sample output will get you some points, while the remaining points will be from hidden inputs that test some edge cases. Think about valid inputs that could break your logic. **Note that the templates given will read from `input.txt` and output to `output.txt`, so it might be a good idea to make a copy of the sample output to avoid overriding it.**

Problem 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,` and `Id` according to https://en.wikipedia.org/wiki/List_of_moments_of_inertia (for a solid cuboid). The program will read the input values of mass, width, height, and depth from the file 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.

Problem 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".

Example inputs:

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

Problem 3: Project Euler Puzzles [8 points]

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.

- Project Euler #1 [2 points], Multiples of 3 or 5
  - Template file: `euler1/euler1.cpp`
- Project Euler #2 [2 points], Even Fibonacci numbers
  - Template file: `euler2/euler2.cpp`
- Project Euler #6 [2 points], Sum square difference
  - Template file: `euler6/euler6.cpp`
- Project Euler #9 [2 points], Special Pythagorean triplet
  - Template file: `euler9/euler9.cpp`

Problem 4: Control Flow Practice [20 points]

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) Factors 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 `1` from to `5`

b) Volume 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) Clip Algorithm [2 points]: If a flag is set (i.e., turned on), clip inputs to be within a given range (inclusive). This program will read input from a file called `clip_input.txt`, each line representing one "test case" in the format `[min_value] [max_value] [value] [clip flag]`. 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 one test case in it, `0 10 15 0`. In this case, the output should be `15` because although 15 is outside of [0,10], the clip flag is set to 0, so the original value is returned (unclipped). You should add more test cases to ensure you code is correct!

   Template file: `clip/clip.cpp`
   Example Inputs: `clip/clip_input.txt`

d) ">10" 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::fstream` 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`

Problem 5: Stroustrup - Chapter 3, Exercise 2 [4 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 significant figs**; this is NOT 3 decimal points necessarily (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` and `0.621371`