***Read all of the following information before starting the exam:***

* The exam is open book, open notes, open Python documentation, open internet, etc.
* You **MAY NOT** use any form of technology to communicate with, send to, or receive information from another person (e.g., classmates, other instructors, anonymous or known persons on the internet). HOWEVER, you are **encouraged** to submit written questions to the professor or TA by email and/or have a private help session with the instructor through ZOOM.
* **MODULES/PACKAGES:** You may now use any Python packages you wish. However, you must follow specific instructions even if a package/library may be the easier way. You may use/reuse code (with proper attribution; e.g., “this function is modified from Dr. Smay’s.py file” or “this import is from my HW1 file”)
* **COMMENTS/DOCUMENTATION:** **ALL** of your functions (especially constructors for classes) should use docstrings and other comments inside the function as necessary.
* **SUBMISSION:** You must place all your .py files in a single folder called EXFSP22 and submit the zipped folder as a single .zip file to Canvas titled EXFSP22.zip.
* **GRADING:** When we grade your assignment, we will run your program with those given numerical values, looking for correct answers. Then we will change the numerical values (including changing the SIZES of the arrays) and look for correct answers for those modified values as well. We will only use numerical values, array sizes and functions that make sense. We will not be testing your program to see how it handles bad data.

1. (30 points) A polymer science question: (main theme tested: OOP, statistics) A file titled polymerClases.py has been uploaded to github for your use. In this file, classes for position, molecule, and macromolecule have been created to model the polymer molecules of poly(ethylene), a common engineering plastic. A poly(ethylene) molecule consists of a string of mers (CH2 in this case) tied together along a backbone of covalent bonds. To estimate the size of a macromolecule, several measures can be calculated such as: *radius of gyration* and *end-to-end distance*.

**You should write** a command line interface (cli) program that asks the user to specify the degree of polymerization (*N*) and the number of polymer molecules to be used in a “freely jointed chain” simulation to calculate average *center of mass* and the average and standard deviations of the *radius of gyration* and *end-to-end distance* for this set of polymer molecules.

Your cli output should look like:

degree of polymerization (1000)?:

How many molecules (50)?:

Metrics for 50 molecules of degree of polymerization = 1000

Avg. Center of Mass (nm) = -23.117, 34.589, 5.621

End-to-end distance (μm):

Average = 0.153

Std. Dev. = 0.006

Radius of gyration (μm):

Average = 196.002

Std. Dev. = 1.583

1. (30 points) An optimization problem: A working, object-oriented program for the quarter car model in MVC has been provided on github. This simulation imagines that the car is traveling at fixed speed along a horizontal plane and encounters a linear ramp up to a new level of y=6 inches. We model the behavior of the car using only a single wheel (i.e., the quarter car model) and treat the tire as a spring between the road and the hub of the wheel and the suspension as a spring and dashpot in parallel between the wheel hub and the car body.

To estimate the spring constants, we guess that a static compression of the spring of the suspension should be between say 3” to 6” and for the tire between 1.5” to 0.75”. Given a car with a mass of 0.25mcar=450kg, use minimize from scipy.optimize to calculate the best combination of k1, c1 and k2 for our car model to minimize the sum of squared errors between the vertical position of the car and the contour of the road.

Diagram

Description automatically generated

1. (40 points) A final gas-power problem. On exam 3, we modified the Otto cycle program to allow for the exploration of the diesel cycle. For this problem, add one more cycle to the program: the dual cycle. Your interface should allow the user to modify all the necessary parameters for the Otto, diesel, or dual cycle. Your GUI should maintain the ability to work in English or Metric units. Your dual cycle should be programmed in a similar MVC pattern to the Otto and diesel cycles.

**About the air standard dual cycle:**

An air standard dual cycle consists of five, internally reversible processes: **1.** An isentropic compression from state 1 (at bottom dead center, V1) to state 2 (at top dead center, V1=r⋅V2), **2.** A constant volume heat addition at top dead center until P3, **3.** A constant pressure heat addition from state 3 to state 4 (P4=P3, V4 = rc⋅V3), **4.** An isentropic expansion from state 4 to state 5 (at bottom dead center, V5=V1), and **5.** A constant volume heat rejection.

An example dual cycle has a compression ratio of r=18, P3/P2 =1.5, a cutoff ratio of rc = 1.2, and a state 1 of T1 = 300K, P1=0.1MPa.