***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 should create a private github repository and invite the TA and instructor as collaborators on this repository. Submit the url to your repository through canvas.
* **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. A diagram of a function

   Description automatically generated(25 pts) The following describes a tiny part of the airplane design process – choosing the engine size needed to meet a take-off distance requirement (STO). STO is the distance an airplane will roll on the runway before it is able to lift off into the air. Calculating STO is performed using a sequence of five equations shown below. Those equations require seven parameters that control takeoff performance. The engine thrust parameter is a major factor in determining STO. The graph shows how STO decreases as engine thrust is increased (all other parameters remaining constant). **Note:** even though units are not explicitly specified, we are working in the English system where gc=32.174(lbmft) /(lbf s2)

Write a GUI-based Python program with line edits for entering the parameters *Weight* and *Thrust*. Your GUI should also have a calculate button that computes and displays (using matplotlib) a graph on the GUI similar to the one shown above with **two differences**: first, your program should produce 3 lines of STO(thrust) (one for the specified weight, one for the specified weight minus 10,000 lb, and one for the specified weight plus 10,000 lb). After the line is displayed, your program should also place a circle on the graph indicating the STO­ at the specified thrust and weight. Your program should be object oriented in the MVC pattern.

1. (25 points) A polymer science question: (main theme tested: OOP, statistics) A file titled polymerClases.py is available 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 a target 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*, *end-to-end distance* and *poly dispersity index* (PDI) for this set of polymer molecules. Note: for PDI to make sense, you should update the polymerClases.py to select the degree of polymerization from a normal distribution with mean=*N* and standard deviation = 0.1*N*.

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

PDI = x.xx

1. (25 points) You have been given access to the quarter car model GUI program. Update this program to compute the forces in the springs and dashpot as a function of time. Display these results on the graph. Note: This graph will be too crowded, so use a tab widget to create a display location for a second graph such that the user can flip back-and-forth between the force vs. time graph and the position vs. time graph.
2. (25 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.