PHYS 162 Fall 2019 Midterm 1

Rules, Guidelines, and Suggestions:

- **Do not begin until you are instructed to do so.**
- No communicating with anyone except for me.
- The only internet resource you can use is the Google Drive folder for this course. You can also refer to *your own* homework files.
- Include your python script(s), not image files, in the zip file that you will submit on Canvas. I need to be able to reproduce your image(s) using your script(s).
- You should produce exactly one python script file for each of the four problems below. Compress all files to be submitted into a single zip file before submitting on Canvas.
- NO CELL PHONES. They cannot be out or visible. You CANNOT use them to keep time and you CANNOT use them as a calculator.
- The test is out of 100 points. Point values for each problem are indicated.
- Be sure to attempt every problem. If you get stuck on a problem, skip it and move on to the next one. Work all the problems you consider easy first. An easy problem and a difficult problem of the same length have the same point value.
- Partial credit will be given assuming you SHOW YOUR WORK and USE COMMENTS to tell me what you're thinking.
- Don't hesitate to raise your hand or come to the front of the room to ask me a question if something is unclear. I can't guarantee I'll be able to give you an answer, but it never hurts to ask.
- You will have until 2:50PM.
- You may keep this copy of the exam when you are done.

1. (25 points) Within Python, use for loops to complete the following tasks:

   (a) (10 points) Find and print to the screen the sum of all the *even* numbers between 37 and 149.

   (b) (15 points) Suppose i is an index that runs over integer values between 8 and 12 (inclusive) and j is another index that runs over integer values between 2 and 6 (inclusive). Create a file called "results.dat" that has three columns: the first has the value of i, the second has the value of j, and the third has the value $e^{(3i-4j)/20}$ with i and j each running over all of their possible values. Be sure that the columns for i and j are formatted as integers and the third column as a floating point number and use formatting codes to make the columns look nice.

2. (20 points) Write a Python script that, when run, creates a directory with a name specified by a command-line argument provided by the user (assume the user will only specify a single word for the directory name; no spaces). To be specific, this argument is read in at the time the script is run; the script itself is non-interactive so it is not permitted to prompt the user for input. Be sure to check that the correct number of command-line arguments are supplied.

3. (15 points) Within Python, define a function of x (using the correct conventions and syntax for doing so) where if x is less than zero, "Negative" is printed to the screen, if x is greater than zero, "Positive" is printed to the screen, and if x is equal to zero, "Zero" is printed to the screen. In none of the cases should anything be returned. Test that your function works as desired. Be sure to use a conditional statement for this problem, not `np.piecewise` or anything analogous.

4. (40 points) Our goal here is to graphically determine (i.e., no numerical solvers allowed!) the solutions to the equation $y_1(x) = y_2(x)$ where $y_1(x)$ is given by some experimental data stored in a data file (provided in the Exams folder on Google Drive; called `exp_data.dat`) and

$$y_2(x) = \frac{(x+20)^{1.1}}{500}\left[9\cos^2\left(\frac{x}{200}\right)+1\right].$$

   Assume that x is given in radians. Write a python script that makes use of `matplotlib` (and other modules as needed) to accomplish the following tasks:

   - reads in the data from the file containing $y_1(x)$ and stores it in one or more arrays.
   - creates a plot of the two functions y_1 and y_2 on the same axes. For this plot:
     - create a title for the plot and label the axes appropriately;
     - use different colors for the two functions;
     - create a legend to identify the two functions;
     - call both `show()` and `savefig()` on your finished product.
   - Use the window that pops up when `show()` is called to graphically determine the intersection points and add solid markers at those points.
   - *Extra credit*: add annotations to the intersection points.