PHYS 361 Lab 3: If statements, loops, and basic plots

Group Warm Up:

The Taylor series for expansion for cos(x) is:

$$cos(x) = 1 - \frac{x^2}{2!} - + \frac{x^4}{4!} - \frac{x^6}{6!} + ... = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

where x is in radians. Write a `while` loop that determines $cos(35^\circ)$. For each pass of the loop, calculate both the series sum and the estimated error:

$$E = \left| \frac{S_n - S_{n-1}}{S_{n-1}} \right|$$

where S_n is the n-th term of the series. Stop adding the terms when $E \leq 0.000001$.

```
clear; %Clear all variables in the workspace
```

Problem 1: Make a unit conversion live script

Write a live script that converts pressure units. At the top of the Pa and psi, or atm and the program will give the equivalent value in another unit system specified by the user. The top of the program should specify (1) the pressure, (2) the orginal unit (either Pa, psi, or atm), and (3) the desired unit (again either Pa, psi, or atm). I have provided two options for getting input from the user to get you started. You can use the built in function `strcmp` to compare to string variables. You do not need to do a fancy formatted output unless you want to.

Use the program to:

(a) Convert 70 psi to Pa

(b) Convert 800 Pa to psi

(c) Convert 1 atm to psi

```
clear; %Clear all variables in the workspace

%Get initial values from user in the command window (option 1)
 %press=input('Input the pressure you want to convert: ');
 %unitin = input('Input the original unit (psi, Pa, or atm):  ','s');
 %unitout = input('Input the desired unit (psi, Pa, or atm):','s');

%Get initial values from user using Live Script features(option 2)
press= 1; unitin="Pa";
unitout="atm";

%Unit Conversion Values
 atm2psi=14.6959;      %Conversion ratio to go from atm to psi, etc.
 atm2pa=101325;
```

```
pa2psi=1.45038e-4;
pa2atm=1/atm2pa;
psi2atm=1/atm2psi;
psi2pa=1/pa2psi;
```

Problem 2: Loop Practice

A person in retirement deposits $300,000 in a saving account that pays 5% interest every year. The person plans to withdraw $25,000 the first year. After the first year, they plan to increase the amount withdrawn by the inflation rate. For example, if the inflation rate is 3% they will withdraw $25,750 the second year. I have added pseudo code in comments to get you started.

a) Determine how many years the money in the account will last using a while loop assuming a constant yearly inflation rate of 2%. Protect your while loop by putting in a counter to prevent it from running indefinitely if you make a mistake. [The correct answer is 16 years.]

b) Determine how many years the money in the account will last using the current inflation rate. Look it up!

```
clear; %Clear all variables in the workspace

%Set initial amount of money

%Use a while loop that runs until the money is less than zero

    %Calculate how much money is withdrawn every year

    %Calculate much money is added to the account due to interest each year

    %Calcuate how the money in the account chabged

    %Increase the year
```

Problem 3: Plotting Functions

According to special relativity the mass *m,* the momentum *p*, and the kinetic energy K of a particle moving with speed *v* are giving in terms of the rest mass m_0 and the speed of light *c* by the equations

$$m = \frac{m_0}{\sqrt{1-\beta^2}}; \qquad p = \frac{m_0 v}{\sqrt{1-\beta^2}}; \qquad K = \frac{m_0 c^2}{\sqrt{1-\beta^2}} - m_0 c^2.$$

where $\beta = \frac{v}{c}$. Using anonymous function and `fplot,` make graphs of $\frac{m}{m_o}$, $\frac{p}{m_0 c}$, and $\frac{K}{m_0 c^2}$ versus β. Choose interesting values to examine and briefly describe the results. You must include labels on the axes that are generated by code for full points.

```
clear; %Clear all variables in the workspace
```

Problem 4: Random Walk

Random walks are useful for simulating the diffusion of particles in a liquid or gas, the movement of bacteria in a solution, or even the escape of photons from the center of the Sun.

First we will program a random walk in one-dimension:

a) Write a program that calculates the number of steps required for a particle to move a distance of ± 10 units. The position x of a particle will be computed by:

$$x_{j+1} = x_j + s$$

where s is a random number between -1 and 1. The initial position of the particle will be $x = 0$. (**Hint**: Use `rand(1)`, which generates a random number between 0 and 1, to generate a random number between -1 and 1. Google "concatenate array MATLAB" if you aren't sure how to add values to the end of an array.)

```
clear; %Clear all variables in the workspace

```

b) Copy your code from part (a) and past it below. Change that code so that it runs through a loop to calculate the number of steps needed to move a length *L* that varies from 5 to 100 units in increments of 5 (`L=10:5:100`). The initial posiiton of the particle will always start at 0.

The code should store the number of steps needed to travel each length, L. This can be done by initializing an empty array (e.g. steps_per_L=[]) at the top of the code and then concatenating new values the the end of the array everytime you run the code for a new length, *L*.

Finally, make a plot that shows the number of steps needed to move each value of L. (e.g. L vs. number of steps).

```
clear; %Clear all variables in the workspace

%Initialize an array to number of steps for each L distance
stepsperL=[];

```

c) **Extra Credit:** Now let's do this calculation in two-dimensions. Instead of having the object move a random distance, the particle will always move 1 unit but the direction of motion will be random. The following code will give you a random number between 0 and 2π. Try running it a few times to confirm.

```
theta=rand(1)*2*pi
```

Once you have selected a random angle you can determine the change in position of a particle in the x and y directions using the following equations:

$$\delta\ x = (1)\cos(\theta)$$
$$\delta\ y = (1)\sin(\theta)$$

Write a program that determines how many steps a particle needs to move a distance of 100 units away from the origin, assuming the particle starts at (x,y)=(0,0). Save the values of x and y for each step and make a plot showing the particle's 2D path through space.

```
clear; %Clear all variables in the workspace
```