CGN 3421, Fall 2018
Homework3
Due: Thursday, Nov. 16, 11PM

Problem 1
Write a Python function `SIRAutoInit` that solves the basic SIR problem in Equations (4.17), (4.18), and (4.19) in the text but automatically sets the parameters β and γ to ensure that outbreaks occur according to Equation (4.29) in the text. The function should solve the SIR problem on the interval [0, T] and return the arrays `S,I,and R` using the statement `return S,I,R`.

Your function should also plot the solutions on one plot with different colors for each array. The horizontal axis should be in units of hours and the vertical axis in units of number of individuals.

Your program will be tested with T = 7000 and N = 350 but you should initially test your program with T = 1400 and N = 4.

```
Inputs:
     Initial Conditions                  S0, I0, and R0.
     Number of Time Steps                N.
     Length of Time Interval (in hours) T.

Outputs:
     numpy arrays representing the solution: S, I, R.
     Plot of the solutions.
```

A function call will have the form:

```
(S,I,R) = SIRAutoInit(S0, I0, R0, N, T)
```

Problem 2.

Write a Python function `SIRTimeRestrict` that solves the Time-Restricted Immunity Problem. Use the same variable names and outputs as in **Problem 1**. However, the function should not set the parameter of the form [β, γ, ν] that holds that parameters to be used.

A function call will have the form:

```
(S,I,R) = SIRTimeRestrict(S0, I0, R0, N, T, Params)
```

Problem 3.

Write a Python function `SimpleWaves` that solves the different equation in Equation (4.42) and plots the numerical solution and the exact solution on the same plot with different colors. The time interval should be [0, 6π/ω]. Your function will be tested with small, medium, and large values of N.

A function call will have the form:

```
x = SimpleWaves(x0,xp0,N,omega)
```

where

```
Inputs:
     Initial Value of x       x0.
     Initial Value of x'      xp0.
     Number of Time Steps     N.
     The value of ω           omega

Outputs:
     numpy array representing the solution:  x.
     Plot of the solution and the exact solution.
```

Problem 4.

Write a Python function `Pendulum` that solves an approximation of the last equation on page 125 by assuming $\theta \approx \sin(\theta)$.

A function call will have the form:

```
Theta = Pendulum(Theta0,ThetaP0,N,Params)
```

where

```
Inputs:
     Initial Value of θ       Theta0.
     Initial Value of θ'      ThetaP0.
     Number of Time Steps     N.
     numpy array of parameters: Params = [m, L, g].

Outputs:
     numpy array representing solution: Theta.
     A plot of the solution.
```