The purpose of this exercise is to learn how to write a basic program in Octave, and to use this skill to develop a program that implements Euler's method for solving an initial value problem. We will be working with the IVP

$$\frac{dy}{dt} = F(t, y), \qquad y(T_i) = Y_i.$$

Recall that Euler's method with step size h produces times t_0, t_1, and so forth and a list of values y_0, y_1, and so forth. The times satisfy $t_0 = T_i$ and $t_{k+1} = t_k + h$ for $k \geq 0$. The values satisfy $y_0 = Y_i$ and $y_k \approx y(t_k)$ for $k > 0$.

At the start of this exercise, you have already downloaded a file `Euler.m` and saved it to a location already known to Octave. In the warmup exercise, you learned how to make modifications to this file. In today's class you will be modifying `Euler.m` to implement Euler's method.

1. Suppose we want to perform Euler's method starting at $t = -1$ and ending at $t = 1$, and that we want to do this in 4 equal time steps h. How long should each time step h be? What are the values of t for each time step? Start with $t_0 = -1$.

2. Suppose we want to perform Euler's method starting at $t = T_i$ and ending at $t = T_f$, and that we want to do this in N equal time steps h. How long should each time step h be?

3. Suppose we want to perform Euler's method starting at $t = T_i$ and ending at $t = T_f$, and that we want to do this in N equal time steps h. Let $t_0 = T_i$, and let t_k in general be the value of t at the k^{th} time step. Write a formula for t_k in terms of T_i, k, and h.

4. Given your formula for h in problem 2 and your formula for t_k in problem 3, what is the value of t_N?

5. If you consider the collection of numbers $\{t_0, t_1, \ldots, t_N\}$, how many numbers are in this collection? Does this agree with your answer in question 1?

6. Ok, now some programming. Modify `Euler.m` so that it takes three inputs: `Ti`, `Tf`, and `N`. The result of calling `Euler(Ti,Tf,N)` should be the vector of times $t_0, t_1, \ldots, t_N$ that would be appropriate for Euler's method over the time interval $[T_i, T_f]$ with N equal time steps. Test that your program works by calling `Euler(-1,1,4)`. Do you get the same answer as in question 1?

Some hints:

- Arrays in Octave are always indexed starting with 1. That is, the first entry of an array `t` is `t(1)`. You will want to return an array `t` with its first entry equal to t_0, its second entry equal to t_1, and so forth. So you want `t(1)`=t_0, `t(2)`=t_1, etc.
- End each assignment line (e.g. `b=Tf-Ti;`) with a semicolon. Otherwise you will see extra output when you run your program.

7. An Octave program can return more than one value. For example,

```
function [a,asquare,acube] = testfunction(x)
  a = x;
  asquare = x*x;
  acube = asquare*x;
end
```

is a function that returns three output variables: it takes a number and returns its value, its square, and its cube. For example:

```
octave-3.4.0:6> [a,b,c]=testfunction(2)
a =  2
b =  4
c =  8
```

Modify `Euler.m` so that it takes three inputs: a function g (which will be a function of one variable), and then the same inputs `Ti`, `Tf`, and `N` as before. It returns two arrays, the first is the array of time values that you have already computed, and the second is an array of values $g(t_k)$ for each k. That is, the second array should be $[g(t_0), g(t_1), \dots, g(t_N)]$.

Test that your new function works by calling

```
octave-3.4.0:11> g=@(x) x.^2
g =

@(x) x .^ 2

octave-3.4.0:12> [t,y]=Euler(g,-1,1,4)
```

Also, try

```
octave-3.4.0:13> [t,y]=Euler(g,-1,1,20);
octave-3.4.0:13> plot(t,y)
```

Does the plot look correct?

8. Modify `Euler.m` so that it takes five inputs: a function `F` (which will be a function of **two** variables), an initial time `Ti`, a final time `Tf`, an initial value `Yi`, and a number `N` of time steps. The result of calling `Euler(F,Ti,Tf,Yi,N)` should be two arrays, one of the t-values and one of the y-values from using Euler's method on the interval $[T_i, T_f]$ with N time steps with initial condition $y(T_i) = Y_i$.

Test that your modification works by using it to find and approximation of

$$y' = y, \qquad y(0) = 1$$

over the time interval $[0, 1]$ with 10 time steps. Be careful when entering the function $F(t, y)$; it's a function of two variables even though it ignores its first variable.

Plot your solution computed with Euler's method and the exact solution at the same time steps all on one plot.

9. Now use Euler's method to compute the approximate solution $y(1)$ with 10, 100, 1000, and 100000 time steps. How many digits of accuracy does each of your answers have?