The *Mathematica* command NDSolve numerically solves a set of ODEs with boundary conditions or initial conditions. Consider this code:

```
theta = Pi / 4;
v0 = 3;
soln = NDSolve[
   {x''[t] == 0,           (* The '' is two single quote marks *)
    y''[t] == -9.8,
    x'[0] == v0 * Cos[theta],
    y'[0] == v0 * Sin[theta],
    x[0] == 0,
    y[0] == 0},
   {x, y},
   {t, 0, 10}
  ];
(* A semicolon at the end of a command suppresses the output for the command *)
```

1) Jeopardy question: The code above is the answer; what physics question is it asking?

Now try it out. To *show* the solution, try adding:

```
ParametricPlot[
 {x[t], y[t]} /. soln,
 {t, 0, 2},
 PlotRange → {{0, 1}, {0, 0.5}},
 AxesLabel → {"x (m)", "y (m)"}  (* The " is a double quote mark *)
]
```

What do you think “ `/.` ” does in this expression?

Plot *y* vs. time. Do your plots (this one, and the parametric one) match what you expect? What should *x* vs. time look like? Check your answer.

2) Consider the following three questions. (Answer them without running any codes yet.)

   A) If you add linear drag, what happens (qualitatively) to the horizontal range (with angle fixed)? Why?

   B) You launch a projectile straight up, first in an ideal (frictionless) world, and then with linear air drag. In which of these cases is the time to reach its *peak height* larger?
   (Or, does it depend? Bear in mind that it won't go as high if there is air drag.) What is your reasoning? (Try to be rigorous!)

C) You launch a projectile at some angle. With linear air drag, does the time to get up to the peak equal the time to fall back down to the ground? (If not, which is larger?) Does your answer depend on the launch angle? Why/why not?

Now, copy your above code, and change the differential equation code to include linear air drag, $\vec{F}^{drag} = -b\vec{v}$.
(Don't forget to add in a value for b in the top line. Do you now need to consider the mass of the object?)
To check for syntax errors, start by setting $b = 0$. You should reproduce the previous plot.
Rerun the "ParametricPlot" with non-zero b to see the new trajectory with drag.

Check your answers to the 3 previous questions you answered, which I summarize as:

A) What happens to the horizontal range?

B) Compare Δt (to peak) in the real and ideal situations.

C) Compare Δt_{up} to Δt_{down}.

Though definitely not needed for qualitative answers, you might find these Mathematica commands helpful, where **t0** *is a numerical value where you want Mathematica to start "hunting" for a numerical solution*:

```
FindRoot[y[t] == 0 /. soln, {t, t0}]
FindMaximum[y[t] /. soln, {t, t0}]
```

Fun stuff: (If you don’t get to it in class, do it at home!!)
Get rid of “everything” *except* the **NDSolve** and **ParametricPlot** command.
Surround the code with the Manipulate command:

```
Manipulate[
    (commands here),   (* Notice the comma at the end of the "commands" *)
 {{theta, Pi / 4, "θ"}, 0, Pi / 2, Appearance → "Labeled"},
 {{v0, 3, "v₀"}, 0, 3, Appearance → "Labeled"},
 {{b, 0, "b"}, 0, 10, Appearance → "Labeled"},
 {{m, 1, "m"}, 0.1, 10, Appearance → "Labeled"}
]
```

Is this cool, or what? The syntax here is that v0 *is initially set to* 3, *but you can vary it from* 0 *to* 3 (*and similarly with the other* 3 *constants*). *Copy it carefully! Try it! This is a great tool for quickly manipulating constants and variables* (*like time*).