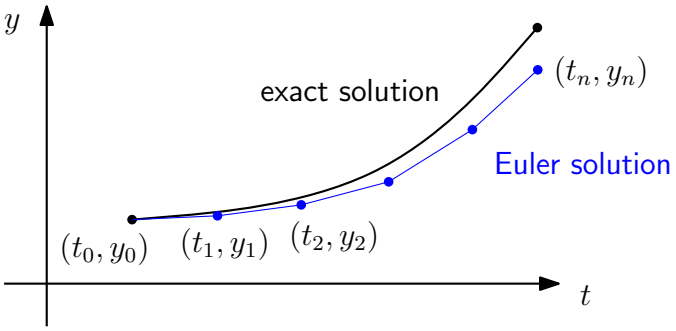
Problem 2.7, #20

In class, we talked about how Euler's method can give us approximate solutions to ODEs. Although Euler's method gives us approximate solutions almost always, we can usually get better and better approximations by decreasing the step size.

The idea of this problem is to test this with the ODE $y' = 1 - t + y$. This ODE we *can* find an exact solution to, and then we will compare the Euler method solution with the exact solution.

The notation used in the problem is that $y(t_0) = y_0$ is the initial condition, and then (t_1, y_1) are the coordinates of the point after the first Euler step, (t_2, y_2) are the coordinates after the second, and so on.



The plan is to use Euler's method (this is part (d)) to approximate the value of the solution at some time t after the initial time t_0 (t is left unspecified, but it's a constant). Then we'll take smaller and smaller steps (which means taking more and more steps to get to t) and see what happens to the approximate value of the solution as we do that.

In part **(a)**, you're asked to first solve the equation exactly, using one of the methods we've learned.

Euler's method starts in part **(b)**. In Euler's method, we take one step at a time, and we get each point (t_k, y_k) using the coordinates of the previous point (t_{k-1}, y_{k-1}). This is like the problem on the midterm that asked you to write down formulas for Euler's method. To get your answer in the form they have, you'll have to do a little bit of algebra.

Part **(c)** is trickier. The problem with the formula in part (b) is that it's recursive: to find y_n, you have to know y_{n-1}, which means finding y_{n-2}, and so on. This will make it tough to see what happens when we take more and more steps. Instead, in this part of the problem we'll try to find a non-recursive formula, by induction.

$y_k = (1+h)y_{k-1} + h - ht_{k-1}$	$y_n = (1+h)^n(y_0 - t_0) + t_n$
old recursive formula in (b)	new non-recursive formula in (c)

The way induction works is that we first show the new formula works for $n = 1$ (called the *base case*). We'll do this by using the old formula (from (b)) for $n = 1$. Then, once we know the new formula works for $n = 1$, we'll try to use this plus the old recursive formula for y_2 to show that the new formula works for $n = 2$. Then we'll combine the new formula for y_2 with the old formula for y_3 to show the new formula works for $n = 3$, and so on. This part is called the *recursive step*. To do the recursive step, we assume that new formula works for y_n, and then combine that with the old formula for y_{n+1} and do some algebra to show that the new formula also works for y_{n+1}. That's your task for part (c).

In part **(d)**, we're finally combining everything together. The idea is to use Euler's method to approximate the solution to the initial value problem $y(t_0) = y_0$ at a time t, taking more and more steps to get there. Here, n will be the number of steps we take. To make it from t_0 to t in n steps, we have to use a step size of $h = (t - t_0)/n$, as the problem says. What you need to do for this part is to plug this step size into the non-recursive formula from (c) and take the limit as $n \to \infty$ to see if, when we let the number of steps go to infinity, we really get the exact solution you found in part (a).