Linearization

One application of differentiation is "linearization;" using the tangent line to a function to approximate values of the function near the point at which the tangent line is calculated. Finding the linearization of $f(x)$ at $x = a$ is equivalent to finding the tangent line to $f(x)$ at $x = a$. You will create a function to help demonstrate linearization and to use it to approximate values.

Create a function called `linearOber` ("Ober" replaced by your own...) in which the function takes a string that is the function $f(x)$, and a value a. Using the commands `sym`, `diff`, `subs`, and `syms`, your function will calculate the tangent line to $y = f(x)$ at the point $x = a$. The function will return the linearization $L(x) = f'(a)(x-a)+f(a)$ (as a symbolic function). In a separate script file that you will publish, you will use your function to do the following problems.

1. Consider the function $f(x) = \sqrt[3]{1+x}$.
   (a) Use your `linearOber` function to find the linearization of $f(x)$ at the point $x = 0$. Careful! You may have to use a different function than `(1+x)^(1/3)` or `nthroot(1+x,3)` in order to make it work (`surd`?). Display the answer.
   (b) Use `linearOber` function to approximate the values of $\sqrt[3]{0.95}$ and $\sqrt[3]{1.1}$.
   (c) Graph the function $y = f(x)$, the tangent line at $x = 0$, and the points corresponding to the approximations of $\sqrt[3]{0.95}$ and $\sqrt[3]{1.1}$ on the same graph. Create several graphs, zooming in to see the difference between the graph and the tangent line at these points. Make sure you create a legend to make things clearer.
2. Let $f(x) = (x-1)^2$, $g(x) = e^{-2x}$, and $h(x) = 1 + \ln(1-2x)$.
   (a) Use your `linearOber` function to find the linearizations of f, g, and h at $a = 0$. What do you notice? Why did this happen?
   (b) Graph f, g, h, and the tangent lines on one graph. Create a legend in order to tell which is which. For which function is the linearization a better approximation (and for approximately what x-values)? Create several graphs, zooming in, to support your answers.

Newton's Method

Another application of derivatives is Newton's Method for finding roots.

Create a function `newtonOber.m` ("Ober" replaced by your own...), that uses Newton's Method to find an approximate solution to the equation $f(x) = 0$, for a given function f. The input should be:

- the function f (given as a string with x as a variable),

- an initial guess x_0 for the solution,
- the desired accuracy (ERROR CHECK: this number should be positive; if not, an appropriate error message should be displayed using the `error` command),
- and the maximum number of iterations allowed (so it will stop if accuracy cannot be reached) (ERROR CHECK: this number should be a positive integer; if not, an appropriate error message should be displayed using the `error` command).

The function uses the Symbolic Math Toolbox (`sym`, `diff`, etc.) to calculate the derivative of the given function f. Your HELP lines should make clear what the inputs are and the order. The output will be the approximation to the solution of $f(x) = 0$. Your code will iterate until the absolute value of the difference between the last two iterations is less than the desired tolerance/accuracy OR the maximum number of iterations has been reached. In either case, **an appropriate message should be printed on the screen** so the user knows if desired accuracy has been reached or not. The message should include how many iterations were completed. Use either `disp` or `fprintf`, and/or `sprintf` for this message; experiment with this. The output of your function is the LAST iterate. In other words, if x_7 was calculated to determine that x_6 is accurate enough, still output x_7 and state that 7 iterations were calculated. If the derivative ever equals 0 at any iteration, an appropriate error message should be displayed and the function will stop (use the `error` command).
The following problems will demonstrate/use Newton's Method. The code for these problems will be in the same published script file as the problems on Linearization.

3. Consider the function $f(x) = x^4 - x - 1$.
   (a) Use your `newtonOber` function to find x_2 using $x_1 = 1$ to find an approximation to a root of $f(x)$.
   (b) Graph $y = f(x)$, the tangent line at $x_1 = 1$, and the tangent line at x_2 to see how the roots of each subsequent tangent line gets closer to the root. Make sure you have a legend and appropriate axes to be able to see everything.
4. Use your function `newtonOber` to find all roots to the equation $e^{\arctan(x)} = \sqrt{x^3 + 1}$ correct to eight decimal places (make sure all decimal places are displayed). In order to do this, first create an appropriate plot to figure out what you are going to use for your initial approximations of x_1 for each root.
5. To demonstrate the importance of that first guess, consider the function $f(x) = x^3 - x - 1$.
   (a) Use your function `newtonOber` to find a root of the equation correct to six decimal places using an initial approximation of $x_1 = 1/\sqrt{3}$. State how many iterations were needed or why Newton's Method didn't work in this case.
   (b) Use your function `newtonOber` to find a root of the equation correct to six decimal places using an initial approximation of $x_1 = 0.57$. State how many iterations were needed or why Newton's Method didn't work in this case.

(c) Use your function `newtonOber` to find a root of the equation correct to six decimal places using an initial approximation of $x_1 = 0.6$. State how many iterations were needed or why Newton's Method didn't work in this case.

(d) Use your function `newtonOber` to find a root of the equation correct to six decimal places using an initial approximation of $x_1 = 1$. State how many iterations were needed or why Newton's Method didn't work in this case.

(e) Graph the function $y = f(x)$, and the tangent lines at $x = 1/\sqrt{3}$, $x = 0.57$, $x = 0.6$, and $x = 1$. Does the graph explain your answers to the above? Make sure your graph has a legend and an appropriate axes (or create a second graph with an appropriate axes) to see what is going on and support your claims.

NOTE: if any of the above return an error, you may need to comment out that code so the rest of your script can run afterwards (keep the code in there to "show your work" and mention something in the comments/text of the webpage).