Due by the 4:00 PM Wednesday, December 17

Instructions: You must work on this project **entirely** on your own.

This project will require you to use some commands that we have not discussed yet. One of them is `fill` and another is `text`. The following requires you to write script files to complete certain tasks. If you also need to write functions files, feel free to do so. The script files should have the first line be a comment that includes your name, Final Project and the task that file is for. The files should also be appropriately commented. This means not a comment for every line, but maybe a comment for a section of related commands. A portion of the grade will not only be on the tasks completed to specification, but whether they are good demonstrations based on some of the choices you made with your code. All script files will make use of `input` and `fprintf`. Format the instructions and output to the screen appropriately, maybe by clearing the command window at first, having line breaks, tabs, etc.

The script files and any other files needed for these tasks should be emailed to me by the due date, with "MA302 FINAL PROJECT" as the subject line.

1. The first task is to create a demonstration of secant lines and tangent lines. The script file should be named `TangentDemo.m`. First, the user is asked to input a vector corresponding to a polynomial. The polynomial should be at least degree 2; check for it. Then the user is asked to input a point the x-coordinate for the point P. Lastly, the user is asked to input the number of graphs, the choices should be 2, 4, 6 and 9 (check for it).

   The script file generates that number of graphs, all displayed using subplot, that shows the polynomial, and the secant line going through point P and another point on the curve, say Q. Have the point Q start at least 1 x unit away from P (use your judgment for the best demonstration - this may depend on how many graphs were chosen) and on the subsequent graphs, have Q approach P. Each subsequent graph has all of the previous secant lines graphed. The last graph should be of the curve, all of the secant lines and the tangent line (in different color) to the curve at the point P.

   Each graph in the subplot should have a descriptive title. Make sure you make the graphs clear and as nice looking as possible; the curve and lines shouldn't be the same color, should there markers for P and the Qs?, is a legend really necessary? etc.

2. This task demonstrates left, right, midpoint Riemann sums and the Trapezoidal rule. The script files should be named `LeftSumDemo.m`, `RightSumDemo.m`, `MidSumDemo.m`, and `TrapezoidDemo.m`. For all of the files, the user is asked to choose between an EXPONENTIAL (E), LOGARITHM (L), SINE (S), COSINE (C), OR POLYNOMIAL (P). If the user chooses polynomial, then the user is asked to input a vector of coefficients corresponding to that polynomial. The polynomial should be at least degree 1, so check for it. Finally, the user is asked to input 2 numbers, a and b, with $a < b$ (check for it). If the user chooses "L", also check to make sure that both a and b are positive numbers.

   Each file will create four plots, all displayed using subplot, of the function the user chose from $x = a$ to $x = b$ (use the standard natural exponential and natural logarithm). Also, in each of the plots you will plot the respective rectangles or trapezoids (using `fill`). Start out with n being small, then increase n such that your choices are appropriate to the demonstration and display them using subplot.

Each of the plots will have a title saying what the function is, and what n is for the graph. In at least one of the titles, it should be clear as to which demo generated the graph; i.e., which Riemann sum or Trapezoid Rule. Also, to make it even clearer you should have all four of them filled with different colors.

3. This task demonstrates Newton's Method. The script file will be called `NewtonDemo.m`. The user is asked to input a vector corresponding to a polynomial of at least degree 3 (check for it!). Then the user is asked for a "first guess" to a root of that polynomial.

   Using the polynomial commands, including `polyder`, you will generate four graphs as subplots. Each of them will have the polynomial graphed, the first guess marked as x_1 just below the x-axis using `text`, and the tangent lines. So the first graph will be the tangent line computed at x_1 = first guess, and the x-intercept of that line will be marked as x_2 just below the x-axis. The second graph will have the tangent line computed at x_2, and the x-intercept of that line will be marked as x_3, etc. If any of these iterations generate a derivative value of 0, no graph is displayed and an appropriate error message is sent to the command window.