Drill Problem: #1

Function Name: heartbeat

Inputs:

1. (*double*) x values for the plot of a patient's heartbeat
2. (*double*) y values for the plot of a patient's heartbeat
3. (*double*) A 1x2 vector of the range describing normal blood pressure

Outputs:

1. (*double*) The area under the plot of the heartbeat
2. (*char*) Blood pressure condition

Function Description:

You are a doctor who has discovered a new way of determining a patient's blood pressure condition from the absolute area under a plot of their heartbeat. This is what you have discovered: depending on certain factors in the patient, there is a range for what the healthy area under a heartbeat plot should be. For demonstration, let's say this range is from x1 to x2. Then let A represent the patient's actual area under their heartbeat plot, calculated with the `trapz()` function:

If `A < x1`, then this patient has `'Low Blood Pressure'`
If `x1 <= A <= x2`, then this patient has `'Normal Blood Pressure'`
If `x2 < A`, then this patient has `'High Blood Pressure'`

Write a function that takes in the x and y data for a plot of a patient's heartbeat, that patient's healthy blood pressure range, and outputs the area under their heartbeat plot and their blood pressure condition. The area under the heartbeat plot should be a scalar double, and the blood pressure condition will be one of the three conditions in apostrophes above: `'Low Blood Pressure'`, `'Normal Blood Pressure'`, `'High Blood Pressure'`.

Notes:

- This function requires the absolute area under the plot. So if there are any "negative areas" below the x-axis, then these will need to be made positive in your calculations.
- Though this function analyzes plot data, and it may help you to look at a plot of the data, there is no output plot of this function.
- This function is not intended to be medically accurate. Do not use this to diagnose your grandmother's blood pressure.

Drill Problem: #2

Function Name: maxGradeMinTime

Inputs:

1. (*double*) A 1xN vector representing times students spent studying for a test
2. (*double*) A 1xN vector of these students corresponding grades on that test

Outputs:

1. (*double*) Time to spend studying for a test
2. (*double*) Predicted grade on said test
3. (*double*) The "coefficient of performance" on the test

Function Description:

You have data for grades of students and how long they spent studying, and you want to figure out the amount of studying time at which the average rate of increase in the student's grade is maximized. This time equates to finding the maximum of the derivative of your input data, or the inflection point on the graph. Use linear interpolation to find where the 2nd derivative of the input data is equal to 0, using numerical integration.

This time to spend studying will then be used to predict the grade earned on the test (output number 2), again using linear interpolation with the input data at the optimal study time.

Finally, estimate the "coefficient of performance": a fictional number that gauges a given person's efficiency in studying based on their study time, resulting grade, and the maximum time any given student spent studying. Use the following equation to evaluate the coefficient of performance:

$$Coefficient\ of\ Performance = \frac{Your\ Grade\ on\ Test}{Your\ Time\ Spent\ Studying}(Max\ time\ in\ data)$$

If the time spent studying was used effectively, then the coefficient of performance will be above 1; if not used efficiently, it will be less than 1.

Notes:

- When taking numerically taking derivatives, always remove the first index from the time vector to accommodate for changing lengths. (Removing the first 2 for the 2nd derivative)

Hints:

- The `interp1()` function may be helpful.
- The `diff()` function may be helpful.

Drill Problem: #3

Function Name: plotShapes

Inputs:

1. (*char*) A string of the shape to be plotted
2. (*double*) The edge/radius length of the shape
3. (*double*) The number of degrees to rotate the shape
4. (*double*) A 1x2 vector representing a translation

Outputs:

(*none*)

Plot Outputs:

1. A plot of the specified shape

Function Description:

Write a function that can plot a square or a circle of a given size; this function should first rotate and translate the shape by the given input specifications, and then plot it. The first input to the function will be a string, either `'square'` or `'circle'`. If the first input is `'square'`, then the second input describes the edge length; if the first input is `'circle'`, then the second input describes the diameter.

The shape should then be rotated clockwise by the number of degrees specified in the third input and translated by the amount designated in the fourth input. The translation input will be given in the form of

```
[x-coord shift, y-coord shift]
```

And should be applied to the entire shape. The object that you plot should be centered at this location. An input of `[0, 0]` as the fourth input should be centered at the origin.

To rotate the shape, use the 2D clockwise rotation matrix from geometry:

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

- Any circle you plot should be composed of 100 evenly spaced points.
- The square should have a point at each corner (no points in between)
- The plot should be made with a **black** line connecting the shape's points.
- Both the x and y axis should go from (- size) to (+ size) where 'size' is the second input.
- The axis should be set to 'square'.

Notes:

- You do not need to worry about the translation putting the object out of the axis range.
- `cosd()` and `sind()` take in degrees instead of radians.
- You will only be given a square or circle to plot.

Hints:

- To rotate pairs of x-y coordinates, separately multiply each pair by the rotation matrix.
- Rotating a circle doesn't change anything.

Drill Problem: #4

Function Name: curveFit

Inputs:

1. (*double*) A 1xN vector of x coordinates
2. (*double*) A 1xN vector of corresponding y coordinates
3. (*double*) The order of polynomial to be fitted to the data

Outputs:

(*none*)

Plot Outputs:

A plot consisting of 4 subplots with the original data, the polynomial fit, the integral and the derivative.

Function Description:

The function will take in data in the form of x and y coordinates and will perform three operations on this data. The first is an n^{th} order polynomial fit of the data, where "n" is the third input to the function. The second is the integral of the polynomial fit, and the third is the derivative of the fit. These are not numerical integrals and derivatives: they should be calculated using the coefficients of the polynomial fit.

Then create a 2x2 subplot of the data. In the first subplot, plot the original input data with the title `'Original Data'`. In the second subplot, plot the polynomial fit with the title `'Polynomial Fit'`. In the third subplot, plot the integral of the fit with the title `'Integral'` and in the last subplot, plot the derivative with the title `'Derivative'`.

When computing points for these functions, you should use the original x data points (input 1) to derive the corresponding y values. When analytically solving for the integral, you may assume the integration constant will always be 0.

Your plots should use all default settings (i.e. you do not need to specify colors, axis range, aspect ratio, etc.).

Notes:

- Iteration is not required to take integrals or derivatives of coefficient vectors.

Drill Problem: #5

Function Name: recursiveCampanile

Inputs:

1. (*double*) The length of the sides of the bottom square
2. (*double*) The rotation angle
3. (*double*) A string of line colors

Outputs:

(*none*)

Plot Outputs:

A plot of a Campanile-like structure

Function Description:

Write a function called `'recursiveCampanile'` that will draw a campanile according to the following parameters:

- The first input is the length of the sides of the square at the base of the campanile.
- The center of the base should be the origin (`x = y = z = 0`).
- The second input will be an angle in radians by which you should rotate each square, after the first, counter-clockwise.
- You will draw the campanile by drawing squares of decreasing size at increasing heights. Each square will have a side length that is 0.9 times the side length of the square below it, and it will be plotted at a distance of 1 above the square below it.
- You should stop plotting squares when the side length falls **below** 1.
- The third input is a string of color characters that you should scroll through each tiem you plot a new square. For example, if the string were `'rbk'`, then the first square would be red, the second would be blue, the third would be black, the fourth would be red again, and so on, repeating until the plot ends.
- Your figure should have the title 'My Campanile', and the x, y, and z axes should be labeled as `'x-axis'`, `'y-axis'`, and `'z-axis'` respectively.
- You **must** use axis equal.
- You **must** call `'view(3)'` at the BEGINNING of your function. Failing to do so will result in your plot being incorrect.

Notes:

- You **must** use recursion for this problem.
- The third input is guaranteed to only contain characters that change the **color** of the lines (no stars, dashes, etc) and can have any number of characters, and the characters may repeat.
- The counter-clockwise rotation matrix is: $\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$.

Hints:

- You may find one or more helper functions extremely useful in solving this problem.