Notice - Testing plot outputs:

For this homework, some of your outputs will be plots. We have provided you with the function `checkPlots()` to help you test the plot outputs of your functions. Use

```
help checkPlots
```

for a full description of how the function works, including what inputs to call with it. Below is the example from the documentation explaining how to run the function.

If you have a function called `testFunc` and the following test case:

```
testFunc(30, true, {'cats', 'dogs'})
```

To check the plot produced by `testFunc` against the solution function `testFunc_soln`, for this test case you would run:

```
[match, desc] = checkPlots('testFunc', 30, true, {'cats', 'dogs'})
```

ALSO!! We are very excited to announce that your lovely TAs have compiled a rockin playlist for you to jam to while completing this homework.

Spotify Version:
https://open.spotify.com/playlist/7wJsC7X6Ff2eYnzt99RbDy?si=6b019c9f853741d8
Converted to Youtube Version:
https://www.youtube.com/playlist?list=PLmcpipPTIAlx2i3r65xGK6F2dDIE1c1fT

Happy Coding!
~Homework Team

Function Name: `kPop`

Inputs:

1. (*struct*) 1xN structure vector containing the heights of different k-pop members

Outputs:

1. (*double*) The standard deviation of the members' heights

Plot Outputs:

1. A scatter plot of all of the members and their heights

Topics: (*numerical methods*), (*plotting*)

Background:

You're listening to your favorite boy bandz playlist on Spotify while doing your CS 1371 homework, when all of a sudden, a song you don't recognize comes on. The song is an absolute banger, so you decide to look up its artist to see who made such a masterpiece and find out that it's a k-pop song! You're overwhelmed with the desire to find out everything about this group, from the members' birthdays to their blood types, but unfortunately, you aren't used to stanning a group with this many members, so you turn to your ult, MATLAB, to help you better visualize the group's stats.

Function Description:

Given a 1xN structure vector with fieldnames `Name` and `Height`, create a scatter plot with the members' heights on the y-axis and the members' positions (indices) in the structure vector on the x-axis. The bounds for the x-axis should range from 1 to N (inclusive). The bounds for the y-axis should range from the minimum height to the maximum height (inclusive). All data points should be plotted as magenta dots with no lines connecting them. Next, calculate the standard deviation of the members' heights and output this value rounded to 2 decimal places. Use this formula to calculate standard deviation:

$$stDev = \sqrt{\frac{\sum(x-\bar{x})^2}{N-1}}$$

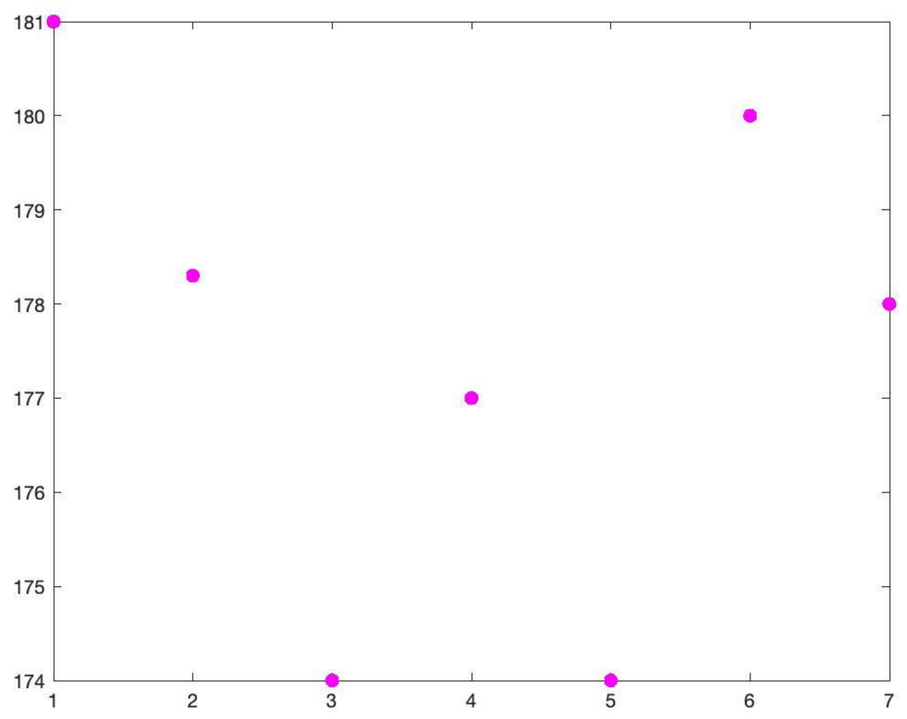
Example:

```
memberStats =
```

Name: Height:	'RM' 181	'Jin' 178.3	'SUGA' 174	'J-Hope ' 177	'Jimin' 174	'V' 180	'Jung Kook' 178

```
stdev = kPop(memberStats);

stdev → 2.7100
```

Plot Output →



Notes:

- All values in the `Height` field are guaranteed to be of class double.
- You do not need to plot standard deviation bars in your figure.
- You are guaranteed distinct minimum and maximum heights.
- There will always be at least two members.
- The size of the dots has been increased for clarity, your dots should be the default size
- N is the length of the inputted structure

Hints:

- Try seeing what happens when you plot y-values without inputting any x-values.
- This problem can be completed without iteration; try not to overcomplicate it!
- Use `help plot` if you are stuck on the point styling.

Function Name: `bigTimeRush`

Inputs:

1. (*double*) A 1xN vector of tour bus velocities
2. (*double*) A 1xN vector of times that correspond to velocity measurements

Plot Outputs:

1. A formatted plot of position, velocity, and acceleration subplots

Topics: (*plotting*), (*numerical calculus*)

Background:

Picture This: it's finally the *Big Night*. *Oh Yeah*! *I Know You Know* you've been waiting for this day *24/seven* for the last year, and now it's time to *Get Up*. "Hurry up we're gonna be late for school," you say to your friend the morning of the *Big Time Rush* concert. The whole day, you daydream like you're *Lost In Love*. Like all your fave fanfics, you hope they wrote a *Song For You*, and that Kendall will ask to be your *Boyfriend* at the end of the night. You're imagining them going *Crazy For U*, when suddenly you get a notification from the fan club that reads: "BREAKING NEWS: Big Time Rush *Worldwide* Tour Bus Stuck in Traffic *Halfway There*, *No Idea* if They Will Make it to Concert in Time". Panicking, you *Run Wild* with your MATLAB skills and tell the other fans they can *Count On You* to calculate, based on the velocity of the tour bus, when the boys will make it to the *City Is Ours*.

Function Description:

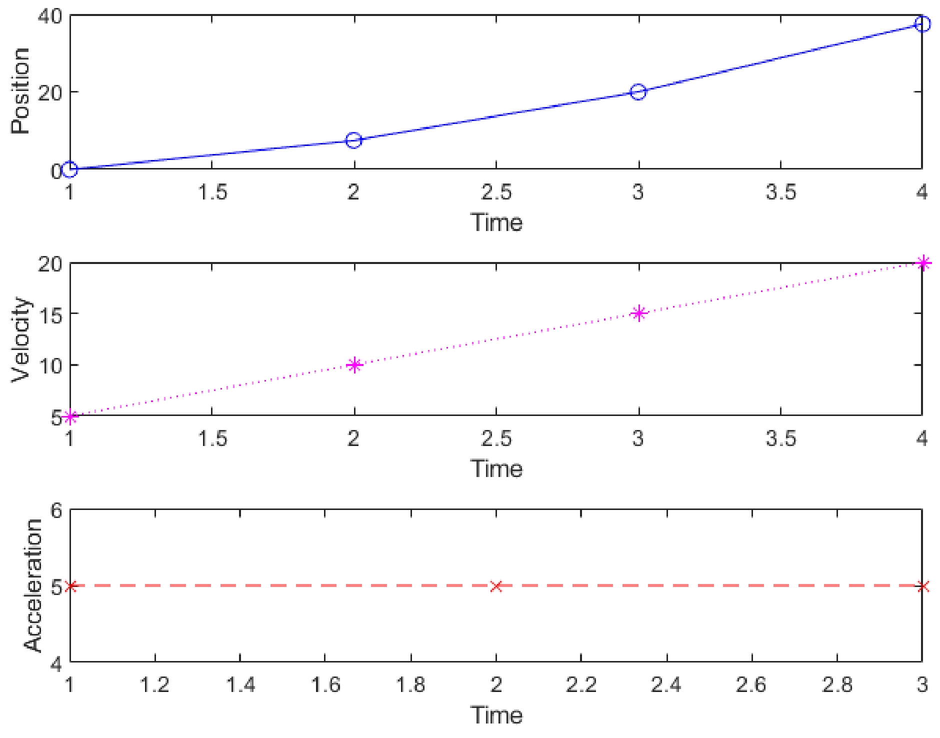
You are given two input vectors of the same length. The first input holds velocity measurements from Big Time Rush's tour bus, and the second input holds time values of the corresponding velocities. Generate a figure of 3x1 subplots with the following values:

1. Subplot 1: The position of the tour bus. Find position by integrating the velocity vector using the cumulative trapezoidal method.
    a. This graph should have a solid blue line with circle points. Label the x-axis `'Time'` and the y-axis `'Position'`.
2. Subplot 2: The velocity of the tour bus.
    a. This graph should have a dotted magenta line with star points. Label the x-axis `'Time'` and the y-axis `'Velocity'`.
3. Subplot 3: The acceleration of the tour bus. Find acceleration by differentiating the velocity vector using the numerical difference method taught in this course.
    a. This graph should have a dashed red line with x-mark points. Label the x-axis `'Time'` and the y-axis `'Acceleration'`.
    b. Using the `diff()` function on a vector of length N produces a vector of length N-1, so make sure you do not include the **final time value** in this subplot.

Example:

```
tourBusVel = [5 10 15 20]
time = [1 2 3 4]

bigTimeRush(tourBusVel, time)
```

Plot Output →



Notes:

- The tour bus's initial position is assumed to be 0.
- Velocity and acceleration may be positive or negative.
- Time will always be positive and in ascending order.
- No additional formatting of your plots is necessary, aside from what is described above.

Function Name: `lodedDiper`

Inputs:

1. (*double*) 1xN vector of X-values
2. (*double*) 1xN vector of Y-values

Plot Outputs:

1. A plot of best-fit curves

Topics: (*curve fitting*), (*iteration*)

Background:

Rodrick and the rest of world-renowned band Löded Diper have been gathering data about their performances all year long. They are a TAD stupid and all of the loud rock music has made them miss a few values here and there. Being the MATLAB wizard and good roadie that you are, you decide to write a function that fills in all the missing holes in the data.

Function Description:

Given vectors of X-values and Y-values, return a plot that includes the given data and sets of increasingly optimized curves. First, plot the given data with red circles and no connecting lines. Then, fit a first order polynomial (straight line) to the data. The y values given by this curve make up the "model y values". Plot this first order polynomial as a solid blue line using 100 equally spaced x values between the minimum and maximum of the original x data. Then, increase the fit polynomial's order by 1 until the r^2 value is greater than 0.9. To calculate the r^2 value, use the formula below. y is the original y data, $\hat{y}$ is the model y values from the previous polynomial fit, and $\bar{y}$ is the average of the original y values.

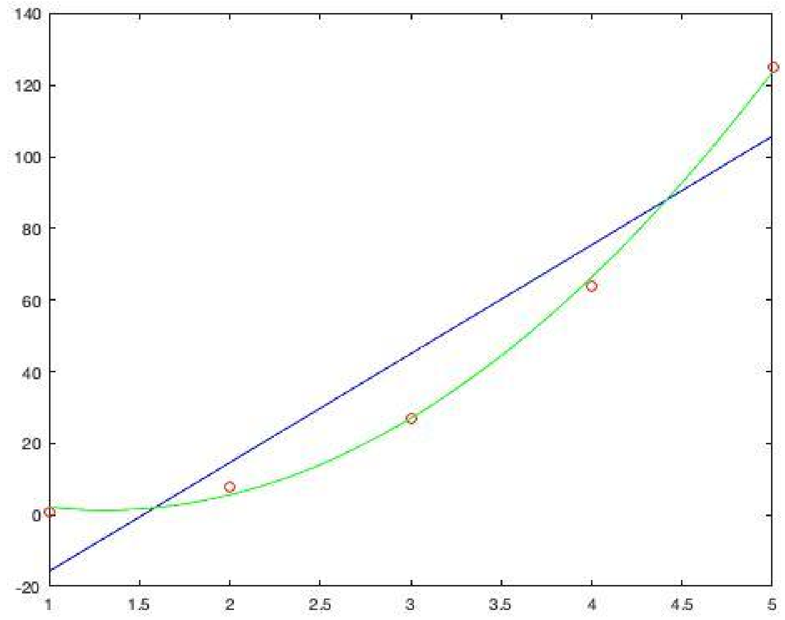
$$r^2 = 1 - \frac{\sum(y-\hat{y})^2}{\sum(y-\bar{y})^2}$$

When r^2 value is greater than 0.9, you can assume you have reached the most optimized curve. Plot this most optimized curve in green.

Example:

```
x = [1 2 3 4 5]
y = [1 8 27 64 125]

lodedDiper(x, y)
```

Plot Output →



Notes:

- The X and Y vectors are guaranteed to be equal in length.
- The green line will always be higher than first order

Hints:

- The functions `polyfit()`, `polyval()`, and `linspace()` may be useful.
- $\hat{y}$ can be calculated with `polyval()`.

Function Name: `stalkTheBoys`

Inputs:

1. (*char*) the name of the .txt file with the corrupted coordinates of your stalkee

Plot Outputs:

1. Figure with a plot containing the data given and and the marked unknown coordinate

Topics: (*low-level I/O*), (*numerical methods*), (*plotting*)

Background:

You are given a text file of all the locations of where your favorite boy band member is going to be when the tour comes to town, but it was corrupted and you can't exactly tell where he will be at any given moment on the schedule. In order to correct this travesty you turn to MATLAB for help!

Function Description:

Given a text file containing the x and y coordinates of band members use MATLAB to interpolate the value of the missing x or y coordinate, represented by a `'?'` in the text file. The text file will contain numerical data for the x-coordinate and y-coordinate **separated by commas**, in that order. There will only be one coordinate pair per line in the text file. There may also be other extraneous characters interspersed throughout all of your data and these characters should be ignored or removed from your numeric data. See the notes for more details.

Interpolate the missing data point using `spline()`, round to the second decimal place, and then plot the data. All of the data points from the original set should be plotted with black diamonds and the point you interpolated for should be represented by a green asterisk on the same plot.

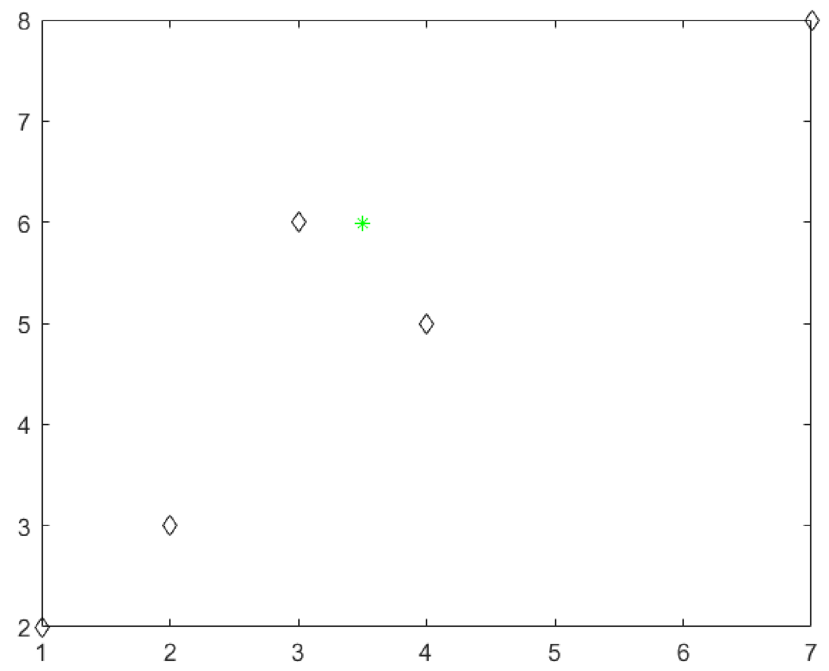
Example:

```
loc.txt:
```

```
1,,2
2,3
4,5i
,3,6
7,8
3.5,?
```

```
file = 'loc.txt'

stalkTheBoys(file)
```

Plot Output →



Notes:

- There may be multiple commas separating your x and y data and multiple leading and trailing commas, but you will never have commas in the middle of a number.
  - ex: `2,3,5` (intended to represent `23,5`) would **not be a valid line** in the file, but `,,,,,1,,,2,,,,` (intended to represent `1,2`) would be valid.
- There will only be one question mark in a file, and it will always indicate the missing coordinate.
- Periods will only be used to indicate decimal values (ex. 3.6).
- Hyphens will only be used to indicate negative values (ex. -4).
- All of the numeric data will be real, so the value *i* will never be used to indicate a complex number and should also be ignored.
  - Likewise, none of the data will be represented in scientific notation, so *e* should also be ignored.
- The data will not contain repeated values for the independent variable, as that would make `spline()` error.

Hints:

- Remember: `strtok()` ignores all preceding delimiters.
- Try removing the extraneous characters before obtaining coordinates.

Function Name: `jonasBrothers`

Inputs:

1. (*struct*) 1x1 structure with the x-y coordinates of each JB in separate fields
2. (*double*) 2xN vector with the x-y coordinates of the fans

Outputs:

1. (*char*) sentence describing how popular the most popular JB is

Plot Outputs:

1. A plot of all the JBs' and fans' locations, colored by who they are a fan of

Banned Functions:

`polyshape, nsidedpoly`

Topics: (*vector/array operations*), (*plotting*)

Background:

Function Description:

Determine which Jonas Brother (JB) is *actually* the most popular and prove this poor lad wrong (or right!). Write a function that takes in the locations of each JB and a bunch of fans, and determines the most popular JB.

The first input is a structure with four fields: `'Joe'`, `'Nick'`, `'Kevin'`, and `'Frankie'`, each containing a 1x2 vector of x-y coordinates (with the format `[x,y]`). The spelling and capitalization of these fieldnames is guaranteed, and there will never be any other fieldnames. The second input is a 2xN double array. In this array, the first row will always contain the x coordinate of each fan, and the second row will always contain the y coordinate of each fan. An individual column in this vector contains the x-y coordinates for a single fan.

The distance between a single fan and each JB determines which one they are a fan of. The JB who is closest (who is the smallest distance away) wins that fan's heart. The distance formula is shown below for your reference.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Count up the number of fans for each JB, and determine who is the most popular. The JB with the most fans is the most popular. Next, determine how many "super fans" that JB has. A super fan is within 10 or less "distance units" from that JB. Output a string describing how popular that JB is, with the following format:

```
'<name of most popular JB> is the most popular Jonas
Brother with a whopping <number of fans> fans and <number
of super fans> SUPER fan(s)!'
```

In addition, plot the locations of the fans, the locations of each JB, and a circle of radius 10 around each JB (the super fan circle). JBs should be plotted as single-point squares, and fans should be plotted as single-point dots. Color each JB and fan as follows:

- Joe, and fans of Joe, should be colored red.
- Nick, and fans of Nick, should be colored green.
- Kevin, and fans of Kevin, should be colored blue.
- Frankie, and fans of Frankie, should be colored black.

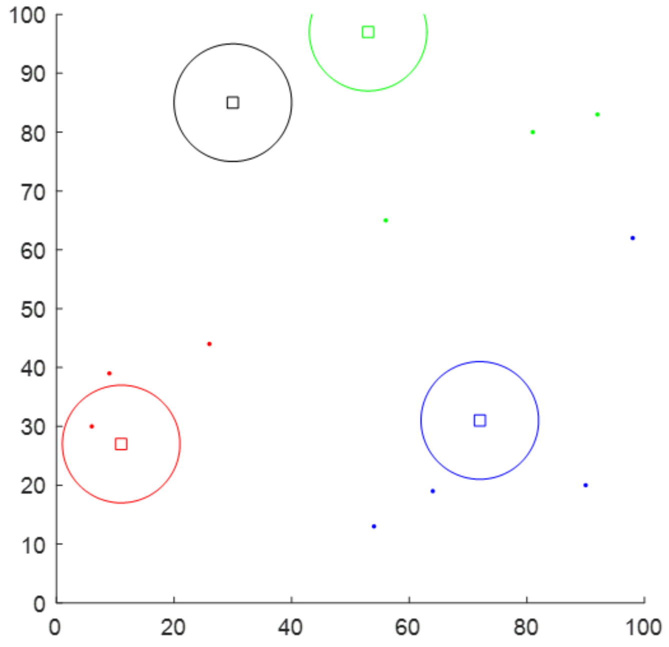
The super fan circle should be solid, match the color of the corresponding JB, and be drawn with 100 points ranging from 0 to 2π, centered on the JB. Finally, make the axes range from 0 to 100 for both the x and y axes and finally use `axis square`. Do these axes modifications in this order.

Example:

```
jonasLocs =

        Joe:  [11, 27]
       Nick:  [53, 97]
      Kevin:  [72, 31]
    Frankie:  [30, 85]


fanLocs = [92 64 26  9 81 56 90  6 54 98
           83 19 44 39 80 65 20 30 13 62]
```

```
popularity = jonasBrothers(jonasLocs, fanLocs)

popularity →
        'Kevin is the most popular Jonas Brother with a whopping 4
        fans and 0 SUPER fan(s)!'
```

Plot Output →



Notes:

- A super fan is still a fan and should be counted as such when determining the most popular JB.
- Always start drawing the super fan circles from angle 0 (directly to the right).
- Don't round distances between fans and JBs.
- There will not be ties in a fan's distance from the JBs.
- The super fan circles may overlap or be cut off on the edges of the plot. This is okay. You do not have to account for this when plotting your circles, plot them as you would normally. There is not an out-of-bounds error for plots.

Hints:

- Take advantage of vector operations to determine distances for entire batches of fans; one of MATLAB's core strengths is the ability to implement vector operations very efficiently, as opposed to iteration. It makes code easier to debug as well!
- The `cos()`, `sin()`, and `linspace()` functions will be useful!