Drill Problem: #1

Function Name: meanEvenOrOdd

Inputs:
1. (*double*) A vector of random positive integers
2. (*char*) A string specifying whether to average the even or odd numbers

Outputs:
1. (*double*) The average of the specified type of numbers

Function Description:

In this problem, you are given a vector of random numbers, and a string of either `'even'` or `'odd'` specifying which type of numbers to average. Using logical indexing, isolate only the even or odd numbers from the vector, and then output their average.

Notes:
- For the sake of this function, treat zero as an even number.
- You are guaranteed to be given the string `'even'` or `'odd'` in all lowercase. This function only needs to accommodate for those two specific strings.

Hints:
- The `mod()` function will prove useful in this function.
- The `mean()` function will also simplify the calculations for this function.

Drill Problem: #2

Function Name: worldSeries

Inputs:

1. (*double*) A 2x9 array of numbers
2. (*cell*) A 1x2 cell with each position representing the name of a baseball team as a string

Outputs:

1. (*char*) The name of the winning baseball team
2. (*double*) The margin of victory of the winning baseball team
3. (*double*) The most number of runs scored in a half inning
4. (*double*) The most number of runs scored in a full inning
5. (*double*) The number of "blank" half innings, where no runs were scored

Function Description:

Baseball has always used a unique scoreboard that displays the number of runs each team scores per each inning. A full inning in baseball is defined by both teams completing a half inning, where their team bats and tries to score as many runs as possible before getting 3 outs. With a traditional baseball game being 9 innings long, the scoreboard of any given baseball game could be stored as a 2x9 double array, where the first row represents one team's runs scored in each half inning and the second row represents the other team's runs scored in each half inning. For example:

```
[ 0 0 1 2 0 2 0 0 1
  1 0 0 3 0 0 0 2 1 ]
```

Write a MATLAB function called `worldSeries` that takes in a 2x9 double array that holds the outcome of a single baseball game and a 1x2 cell array that holds the names of two baseball teams. Your function needs to determine the following information for the [5] outputs:

1. The name of the winning team is determined from whichever row in the given array has a greater sum. The first row is for the first team in the cell array, and the second row for the second team.

2. The margin of victory of that team (guaranteed to be greater than 0)

3. The most number of runs scored in a half inning – i.e. the maximum value in the whole array.

4. The most number of runs scored in a full inning – i.e. the maximum value of any column in the whole array.

5. The number of "blank" half innings is the total number of 0s that appear in the array.

Notes:

- There are guaranteed to be no instances of a tie for the final outcome; therefore, your first input is guaranteed to be a 2x9 double array (no extra innings) and your output for the margin of victory is guaranteed to be greater than 0.
- Don't worry about the instance of a repeated maximum value: your outputs should be the value itself, regardless of which inning/half inning it occurred in.
- All "blank" half innings will have a value of 0.
- For the astute baseball fan: don't worry about the instance of the home team not needing to bat in the bottom of the 9th inning if they're winning. The value will be stored as 0 in that case, so you'll need to include it in your count of "blank" half innings.

Drill Problem: #3

Function Name: rainfall

Inputs:

1. (*double*) The size of a single raindrop, in millimeters
2. (*double*) The starting height, in meters, of a single raindrop

Outputs:

1. (*double*) The final size of the raindrop, in millimeters, once it reaches the ground

Function Description:

Ugh…it's raining? Again? This is the worst…Let's write a MATLAB function about the phenomenon of rainfall and why it keeps trying to ruin our days.

Write a MATLAB function called `rainfall` that tracks the life of a single raindrop from a given starting height to the ground. The inputs are both given as doubles, with the first referring to the starting raindrop size (in millimeters) and the second referring to the starting height of the raindrop (in meters.)

This function will **recursively** track the size of a raindrop for every 50 meters it falls. A single raindrop is held to the following idealized case:

1. For every 50 meters that the raindrop falls, it will collide with another raindrop and become 1.5 times its original size.
2. Once a raindrop reaches 4mm or greater in size, it will evenly split back into two raindrops.

This function should track the raindrop's size for every 50 meters it falls and then output its final size one it reaches the ground, i.e. once its height is less than or equal to 0.

Notes:

- The function `rainfall` MUST be recursive to receive credit.
- If a raindrop has reached the ground level with a size greater than 4mm, it should remain that size and not split a final time.
- On a recursive call that requires the raindrop to split, the function only needs to keep track of one of those raindrops, not both of them.

Drill Problem: #4

Function Name: punchLine

Inputs:

1. (*char*) The name of a .txt file of a conversation between chickens

Outputs:

1. (*char*) The proper punchline depending on which of three key words appears most often in the .txt document

Function Description:

Why did the chicken cross the road? Was it to get to the other side? Was it??

Some chickens have recently been chatting and a Jokester is on the loose. He's been interrupting conversations and interjecting this age-old joke, but he leaves before he gives the answer! Your job is to read through the chickens' conversation and determine which punchline the Jokester was going to say.

The chickens reason that the Jokester only calls out one of three punchlines based on the following key words:

Key word	**Punchline**
`side`	`To get to the other side!`
`duck`	`To get away from the duck!`
`girl`	`To see about a girl!`

Write a MATLAB function called `punchLine` that will return the proper punchline of the Jokester based on whichever of the three key words appears MOST often in the .txt file given. The key words will appear exactly as is (all lowercase) and there will only be one key word that appears more often than the other two.

Notes:

- The .txt file for the chickens' conversation may have any number of lines.
- You do not need to account for any extra capitalization; if the key words do not appear in all lowercase, then they should not count.
- There are guaranteed to be no ties between the counts of key words.

Drill Problem: #5

Function Name: defeatZurg

Inputs:

1. (*char)* The name of an image of the evil Emperor Zurg
2. (*double*) A 2x6 array of laser beam locations

Outputs:

1. (*char*) The outcome of the showdown with Emperor Zurg

Output Files:

1. A displayed figure containing the original image of Zurg with the laser beam locations plotted as red circles

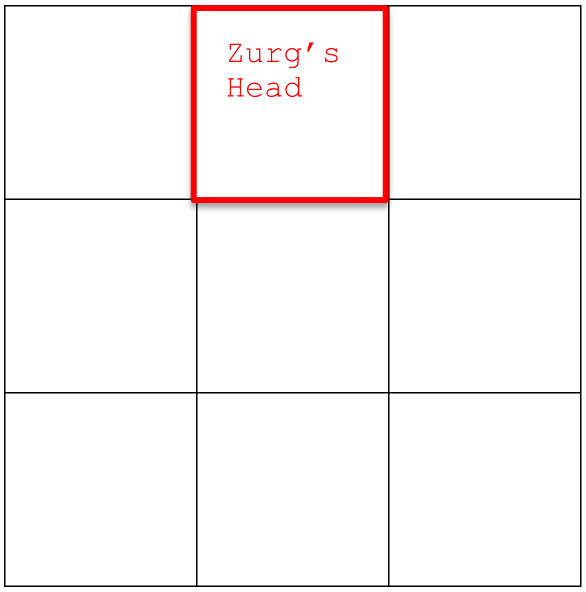
Function Description:

"Mayday! Mayday! Come in Star Command! Send reinforcements! Star Command do you copy?"

Oh no! Intergalactic Space Ranger Buzz Lightyear has crash landed and run into his arch nemesis, Emperor Zurg. Zurg begins to load up his tri-ion blaster, but Buzz, equipped with his trusty red wrist laser, needs to shoot down Zurg and send his health to 0 in order to save himself! Once Zurg has fired two rounds of his tri-ion blaster (i.e. 6 times) it'll be too late, so Buzz only has time to fire 6 laser beams.

Given an image of evil Zurg, write a MATLAB function called `defeatZurg` that takes a 2x6 array of laser beam locations and determines whether or not Zurg can be defeated with the given data. The first row of the array defines the x coordinates of the laser beams, while the second row of the array defines the y coordinates. Zurg starts with a health of 100 and his health decreases by 30 for each laser beam that strikes him in the head and by 10 for each laser beam that strikes him in the body.

The area for Zurg's head is considered to be the top third of the image with respect to the y axis and the middle third with respect to the x axis. Refer to the following figure:



Drill Problem: #5

If a given laser beam falls within the area of Zurg's head, his health decreases by 30. When hit anywhere else with a laser beam, his health will only decrease by 10.

The function ought to plot the location of each laser beam onto the given image of Zurg with a red circle and display that in a new figure once the function has ended. The function output needs to be a string indicating whether or not Zurg was defeated. If Zurg's health reaches 0 or less, the function should end and output: `'YOU DEFEATED ZURG!'` If Zurg's health does not reach 0, the function should output: `'GAME OVER!'`

Notes:

- If a laser beam happens to hit exactly on the bounds for Zurg's head, it should count as a head shot.
- Note that the coordinates for the laser beams start with an origin of (0, 0) in the upper left-hand corner of the image.
- Your code does NOT need to call `figure` separately.
- Once an image is read in, if you call the command `image(imgMatrix)`, call `hold on`, and then plot some x/y data, the plotting will appear on the image display itself; that is the intention behind this problem.
  i.e. your first two lines of code should look like:
  ```
  A = imread(input);
  image(A);
  hold on
  ```
  >> Then you can call plot afterwards and the red circles will appear on the image.

Drill Problem: #6

Function Name: highRise

Inputs:

1. (*double*) A Nx2 array of heights and widths for building profiles

Outputs:

(*none*)

Plots:

1. A single plot containing the 2D building profiles of the input array

Function Description:

Remember the previous homework problem `flood` from week 11?? Well, now that we know all about plotting, let's plot the 2D profile of those buildings, but in a little bit different scenario…hopefully one that doesn't require too much…pain?

Write a MATLAB function called `highRise` that takes in an Nx2 array of points where the first column defines each building's height and the second column defines each building's corresponding width. The objective is to plot the rectangular profiles of the buildings side by side, where the first row in the given array defines the height and width for the first building, the second row defines the second building, and so on until all the buildings have been accounted for. There are a few rules for plotting the buildings:

1. The bottom left corner of the first building will be the point (0, 0).

2. The bottom left corner of the next building will be the point (0 + building1width + 1, 0) i.e. **the width of the previous building + 1**.

3. The **colors of each building will alternate**, starting with *blue* followed by *black*, then blue, then black, etc.

Notes:

- There may be any number of buildings given in the input array.
- All buildings should be plotted onto a single figure.
- You do NOT need to manipulate the axes of your plot at all.
- The heights and widths are arbitrary values and not properly scaled to be actual building profiles.
- Note the *additional spacing* of the x coordinate (+1) each time a new building is plotted; this part differs from what was described in the previous homework problem, `flood`.

Drill Problem: #7

Function Name: dataRead

Inputs:

1. (*char*) A string containing the name of a .txt file of data (in CSV format)

Outputs:

1. (*struct*) A structure containing the data read from the input file

Function Description:

One of the most common outputs for data gathering devices is a text file with the file's information at the start and the data that was collected in the following lines separated into columns by commas. This format is called a CSV (comma separated values) file and while it is easy for a machine to write data into this format, it is rather unwieldy to work directly with a text file.

Write a MATLAB function called `dataRead` that takes in a text file of data in CSV format and outputs a 1x1 structure containing that data. The fieldnames of the structure will be listed in the first line of the file and the contents of each field should be a vector of the numerical data taken from the column corresponding to the field. For example, a file containing:

```
time,temperature,force,
1.00,50.75,107.90,
1.50,45.64,109.12,
2.00,43.83,112.52,
2.50,41.57,115.82,
3.00,38.41,118.90,
```

Would be formatted into the following 1x1 structure:

```
time:           [1.00  1.50  2.00  2.50  3.00]
temperature:    [50.75  45.64  43.83  41.57  38.41]
force:          [107.90  109.12  112.52  115.82  118.90]
```

Notes:

- Commas will only be used to separate data and there will be a comma included at the end of every line.
- The class of each field needs to be `'double'`.
- The values in the output structure's fields should be **column** vectors.
-

Drill Problem: #8

Function Name: rollerCoaster

Inputs:

1. (*char*) The name of an excel file containing the roller coaster data for a theme park
2. (*char*) The name of a category to search for
3. (*char*) A string indicating whether to search for the most or least value in the category

Outputs:

1. (*char*) The name of the roller coaster ride to go ride

Function Description:

You've just been hired as the Operational Manager of an established theme park in the world of Roller Coaster Tycoon. You are learning the laws of the land and find that your customers often spend entirely too much time walking around your park looking for roller coasters to ride. Even with their trusty Park Maps in pocket, you want to do your best to help them out, so you decide to use MATLAB to help them answer the "which roller coaster should I ride?" question.

Write a MATLAB function called `rollerCoaster` that takes in the category a customer wants to make a decision off of and whether they want the "most" or "least" of that category. The data for the theme park is given in an excel file and might look like the following:

Ride	Wait	Thrill	Fear	Cost
Goliath	5	7.5	8.4	3
The Demonator	9	9.9	7.2	4
Corkscrew	4	6.4	6.1	2.50

For this example, if a customer wanted to find the roller coaster with the **least cost**, they would find `'Corkscrew'` to be the output from the `rollerCoaster` function.

The customer can choose from any of the column headers as their second input for the category (other than the `'Ride'` column, of course). The third input will either be the string `'Most'` or `'Least'` depending on what circumstance they seek. If a customer is unable to make up his or her mind, they might use some other string for the third input. If that is the case, your output should read: `'Just pick one!'` and you will politely move on to help out the next customer.

Notes:

- The first row of the excel file is guaranteed to be the headers for each of the columns; the excel file may have <u>any number of categories in any order</u>, but the rides will always be listed in the first column, titled `'Ride'`.
- All of the data for each ride is given as data type double.
- The second input is guaranteed to be the name of one of the categories listed in the column headers.
- If the third input is anything besides the string `'Most'` or `'Least'` (capitalization does NOT matter), then your output should be: `'Just pick one!'`

Drill Problem: #9

Function Name: bondHackBond

Inputs:

1. (*char*) The name of a .txt file from a secret database

Outputs:

1. (*char*) The 6-letter password to hack the door

Function Description:

The name's Bond…James Bond.

Going rogue on another secret mission, Bond finds himself at a vaulted door that is protected by a secret keypad. Bond activates the screen and discovers a 6-letter password is required to unlock the vaulted door.

Bond contacts M back at MI6 and M uses a secret database that generates a .txt file with the correct 6-letter password to unlock the door. The .txt file, however, displays all of the guesses that the secret database used, so an excerpt of it looks something like this:

```
SAILED
INSERT
VIKING
VLIGHT
VPLACE
TARGET
```

The pattern for selecting the correct character for one of the 6-letters in the password is to find the character that appears 3 consecutive times in the same position of the line from the .txt file. (You'll notice the 3 consecutive 'V's in the example above in the first position, so 'V' is the first letter of the 6-letter password to unlock the door).

Write a MATLAB function called `bondHackBond` that finds the correct 6-letter password from a given .txt file to unlock the vaulted door. The given .txt file has an indeterminate number of lines, but it is guaranteed that a single character will repeat exactly 3 consecutive times for a given position of the password. The function ought to read through the .txt file, capture each correct character of the password, and output the 6-letter password so that Bond can continue his secret mission.

Notes:

- The database that generates the .txt file always finds the correct character of the password with 3 consecutive identical letters. It will then search for the next position of the password and will NOT find another pattern of 3 in the previous position.
- The password is guaranteed to be 6-letters in length; the letters are guaranteed to be capital letters 'A' through 'Z'.

Hints:

- Consider finding a way to read through the text file over and over again, while during each iteration, you find the next correct character for the output password.

Drill Problem: #10

Function Name: cyberChase

Inputs:

1. (*struct*) A 1x1 structure with 10 nested structures that represent the 10 Levels of the Scooby Doo Game: Cyber Chase

Outputs:

1. (*char*) The outcome of what level in the game is reached
2. (*double*) The total number of monsters defeated
3. (*logical*) True or false indicating whether or not the game is won

Function Description:

Zoinks! Mystery Inc. is at it again and have found themselves searching through a computer game called Cyber Chase in order to catch and defeat the evil Phantom Virus. Mystery Inc. has asked you to help them on their adventure: they have each level of the Cyber Chase game stored in a structure and they need you to use MATLAB to determine several pieces of information for them.

Write a MATLAB function called `cyberChase` that takes in a 1x1 structure. The given structure has the following fields:

Field	**Description**
Level	(*double*) Current level number
Monsters	(*double*) Number of monsters on level
Gang	(*cell*) Cell of strings indicating characters on level
Item	(*string*) Name of item to collect on level
ScoobySnax	(*logical*) True or false indicating if Scooby Snacks are found on level
Next	(*struct*) Next level to go to, stored as structure ***on level 10, this field will be empty

The only way to reach the next level is by checking the ScoobySnax field of each level. If the field has a value of true, then you can advance to the next level which will be a structure of the same format stored in the Next field. The intermediate levels may involve any of the original Mystery Inc. characters as well as their "Cyber" counterparts. For example, you might see `'Shaggy'` and `'Cyber Shaggy'` as strings within the Gang field.

The only way to win the game Cyber Chase is by weakening the Phantom Virus on the 10th and final level with the item: `Magnet`. This can only be achieved by having both Scooby and Cyber Scooby in the Gang field on the 10th level.

For each level of the game, you will need to do the following:

1. Store the current level number.
2. Add the total of monsters defeated from the Monsters field.
3. Collect and store a list of items from the Item field.
4. Check the logical value of the ScoobySnax field.
5. Obtain the next structure to go through from the Next field.

The three outputs for the function are as follows:

Output	**Description**
outcome	Format: `'You reached level: %d'` where %d indicates the level number reached
monsters	Total number of monsters defeated from all levels reached
log	True if following conditions are met: - reach level 10 - Gang on level 10 includes characters: `'Scooby'` and `'Cyber Scooby'` - One of items collected is: `'Magnet'`

Notes:

- Once you've reached the 10th level there are 3 things to consider:
  - Is `'Scooby'` in the Gang field?
  - Is `'Cyber Scooby'` in the Gang field?
  - Is `'Magnet'` one of the items collected?

    If these three conditions are met, your third output should be true. If at least one of them is not met, your third output will be false.
- Even if the ScoobySnax field is false, the count for monsters on that level should add to the total.
- There is guaranteed to only be one item in the Item field of each level.
- The only existence of an empty field will be the value of the Next field on the 10th level. All other fields will be populated as specified.

Hints:

- The function should stop if and when the ScoobySnax field is false. You might consider that part of the terminating condition for a well-constructed while loop.
- The function should also stop if and when you've looked through the 10th level (i.e. the value of the Next field is not another structure).

Drill Problem: #11

Function Name: recursiveCampanile

Inputs:

1. (*double*) The length of the sides of the bottom square
2. (*double*) The rotation angle, in radians
3. (*char*) 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.