General Instructions

Due Date:
Saturday, March 4 at 11:00pm (submit via blackboard)

Assignment Summary Instructions:
This assignment has three problems summarized below. You will use MATLAB as a tool to solve the problems for the given test cases provided and that it is flexible for any additional test case that might be used to evaluate your code.

- Problem #1: Problem Solving for Equivalent Resistance
- Problem #2: Problem Solving for Data Analytics Tool
- Problem #3: Application to Inventory Simulations

Submission Instructions:
You will submit a **.zip file** that will contain the following files:

1. **.m file – Matlab file (separate sections using %% for each problem)**. Your final script should be in the zipped folder and named as **MA#_USERNAME.m (for example, MA5_dwburles.m).**
2. .pdf file - scan of your algorithm sheet (use the template) for Problem #1
3. .pdf file - scan of your algorithm sheet (use the template) for Problem #2
4. .pdf file - scan of your algorithm sheet (use the template) for Problem #3

Submit your **.zip file** to blackboard using the Mastery Assignment 1 link. Your final submission should be a zipped folder named as **MA#_USERNAME.zip (for example, MA5_dwburles.zip).** The file must be completely submit before 11pm on March 4 for credit. **No late work is accepted and will result in a zero on the assignment**. It is your responsibility to make sure the file is completely submitted prior to the deadline. You are provided 2 upload opportunities in case your first upload is incomplete or a mistake.

Academic Honesty Reminder:

The work you submit for this assignment should be your work alone. You are encouraged to support one another through collaboration in brainstorming approaches to the problem and troubleshooting. **However, all support should be only verbal in nature**. Sharing files and showing other students your file is considered physical and visual help and is considered an academic honesty violation.

Some examples of academic misconduct in ENGI 1331 include but are not limited to the following actions:

1. Picking up and using or discarding another student's written or computer output;
2. Using the computer account of another student;
3. Representing as one's own the work of another on assignments, quizzes, and projects;
4. Giving another student a copy of one's work on an assignment before the due date.
5. Copying work from online resources (Chegg, google forums, etc.)
6. Posting work to online resources where other students can view your work

This assignment will be checked for similarity using a MATLAB code. The similarity code will check each submission for likeness between other student submissions, past student submissions, the solution manual, and online resources and postings. If your submission is flagged for a high level of similarity, the ENGI 1331 faculty will review the files, and then the guilty parties will be turned in for an academic honesty violation if deemed appropriate.

NOTE: Since this is an automated system for all sections, if any of your work is not your own, you will be caught. Changing variable names, adding comments, or spacing will not trick the similarity algorithm and will result in a violation.

Problem #1

Background: Electrical resistors are said to be connected "in series" if the same current passes through each and "in parrellel" if the same voltage is applied across each. If in series, they are equivalent to a single resistor whose resistance is given by equation (1). An example of resistors in series are shown in the figure next to the equation. If the in parallel, their equivalent resistance is given by equation (2) with an example of resistors in parallel shown in the figure next to the equation.

$$R_{eq} = R_1 + R_2 + R_3 + \cdots R_N \qquad (1)$$

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \cdots + \frac{1}{R_n} \qquad (2)$$

Your program should prompt the user to enter a vector of resistances in ohms then prompt the user to select the type of connection (series or parallel). Then calculate the equivalent resistance (R_{eq}). The program should then ask the user if they would like to enter another set of resistors. If the user selects yes, repeat the program. Otherwise, your program should end.

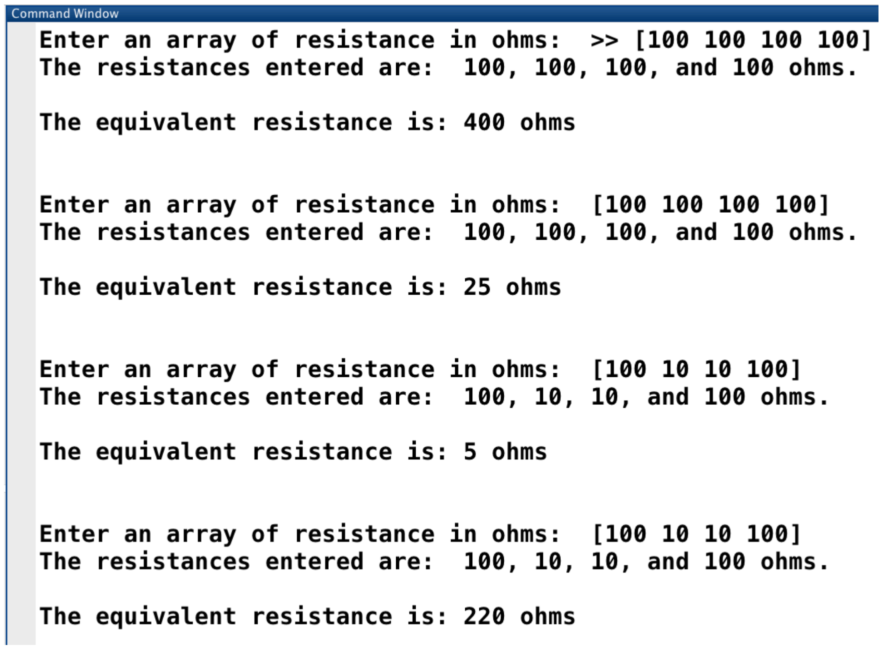
Algorithm Requirements:

- Create an updated algorithm with the modifications described above. It can be typed or by hand.
- Save or scan the file to a .pdf file and include in your final .zip submission

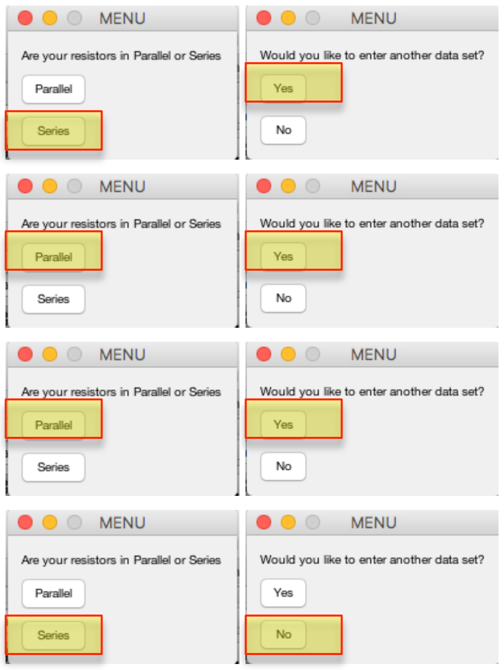
Coding Requirements:

- Use For Loops for data processing and calculation
- Use While Loops for controls
- User input for resistance
- User menu for type of connection
- User menu for repeat of program
- Formatted output as shown in the Sample Output below.

Sample Output to Command Window



```
Command Window
Enter an array of resistance in ohms:  >> [100 100 100 100]
The resistances entered are:  100, 100, 100, and 100 ohms.

The equivalent resistance is: 400 ohms


Enter an array of resistance in ohms:  [100 100 100 100]
The resistances entered are:  100, 100, 100, and 100 ohms.

The equivalent resistance is: 25 ohms


Enter an array of resistance in ohms:  [100 10 10 100]
The resistances entered are:  100, 10, 10, and 100 ohms.

The equivalent resistance is: 5 ohms


Enter an array of resistance in ohms:  [100 10 10 100]
The resistances entered are:  100, 10, 10, and 100 ohms.

The equivalent resistance is: 220 ohms
```


MENU
Are your resistors in Parallel or Series
Parallel [✓ Series]
MENU
Would you like to enter another data set?
[✓ Yes] No
MENU
Are your resistors in Parallel or Series
[✓ Parallel] Series
MENU
Would you like to enter another data set?
[✓ Yes] No
MENU
Are your resistors in Parallel or Series
[✓ Parallel] Series
MENU
Would you like to enter another data set?
[✓ Yes] No
MENU
Are your resistors in Parallel or Series
Parallel [✓ Series]
MENU
Would you like to enter another data set?
Yes [✓ No]

Problem #2

***Background**: You are working for a data analytics firm that has been asked to create a generic data collection tool that will provide basic statistics on input typed in to the screen. This data collection tool should ask the user to type the number of desired data points they need to record, the number of decimal places for all of the numbers displayed in the final output, then the program should allow them to record all of the data points into the Command Window, one by one, where the user presses the Enter key after each data point. You may assume that this program will only need to handle numeric values and will not need to worry about strings, vectors, or input of other variable types.*

As soon as the user inputs all of the values, your program should generate a string representation of the vector of data, as well as provide basic statistics including the number of negative values, number of positive values, the sum of all of the values, the mean, median, and standard deviation of the data set, and the minimum and maximum values in the data set. The output should appear similar to the output shown below. Note that the vector of data does not need to be displayed with the number of decimal places specified by the user.

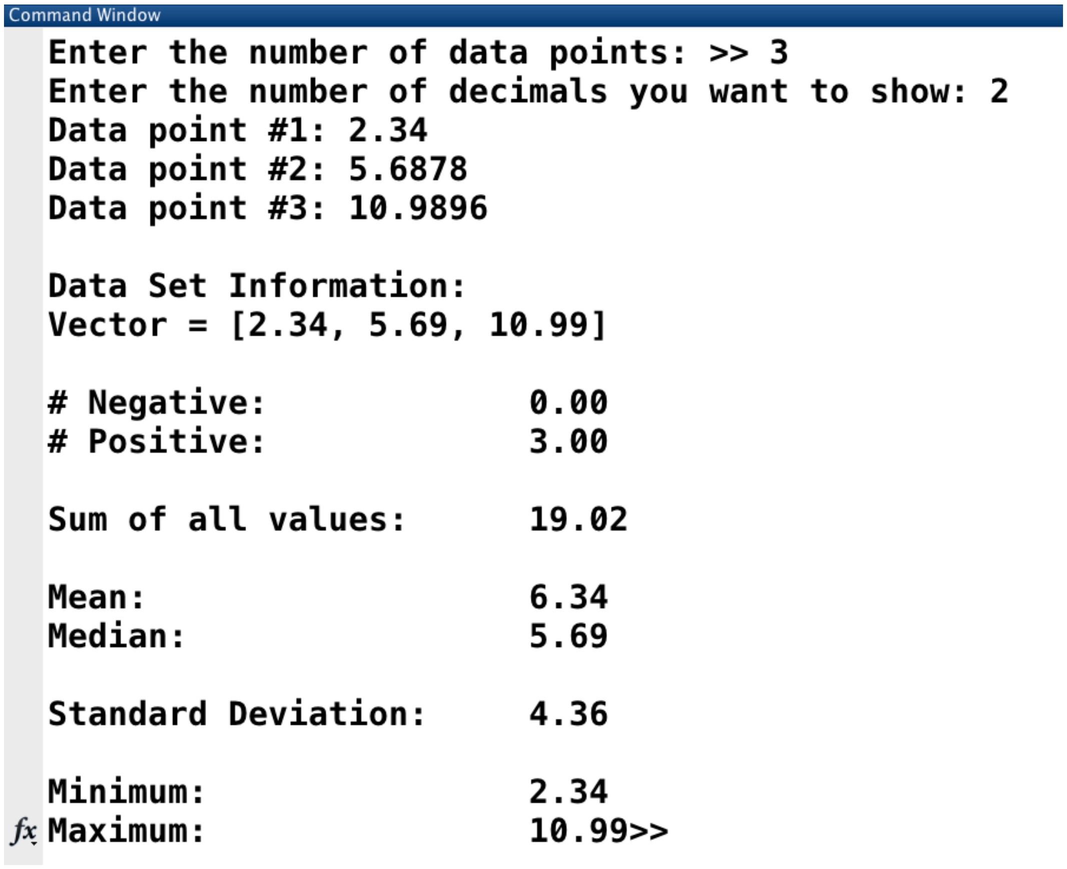
Algorithm Requirements:

- Create an updated algorithm with the modifications described above. It can be typed or by hand.
- Save or scan the file to a .pdf file and include in your final .zip submission

Coding Requirements:

- Use For Loops
- The vector of data does not need to be displayed with the number of decimal places specified by the user.

Sample output to Command Window
(You code should handle any number of data points. You should create your own test cases for additional testing)



```
Enter the number of data points: >> 3
Enter the number of decimals you want to show: 2
Data point #1: 2.34
Data point #2: 5.6878
Data point #3: 10.9896

Data Set Information:
Vector = [2.34, 5.69, 10.99]

# Negative:             0.00
# Positive:             3.00

Sum of all values:      19.02

Mean:                   6.34
Median:                 5.69

Standard Deviation:     4.36

Minimum:                2.34
Maximum:                10.99>>
```

Problem #3

Background: A certain company manufactures and sells golf carts. At the end of each week, the company transfers the carts produced that week into storage (inventory). All carts that are sold are taken from the inventory. A simple model of this process is given in the following equation:

$$I(k+1) = P(k) + I(k) - S(k)$$

where,

P(k) = number of carts produced in week k
I(k) = number of carts in inventory in week k
S(k) = number of carts sold in week k

The projected weekly sales for 10 weeks are shown as an example in the table below.

Week	1	2	3	4	5	6	7	8	9	10
Sales	50	55	60	70	70	75	80	80	90	55

Suppose the weekly production is based on previous week's sales. So, for this example, assume that the first week's production is equal to the initial inventory of 50 carts; that is, P(1) = 50.

The goal of your program is to ask the user to enter an initial inventory and a vector of sales (various lengths). Then compute the number of carts in inventory for each of the weeks in the vector provided or until the inventory drops below zero. You should produce the table (for each week) up to the before inventory goes below zero.

EXTRA CREDIT: Create a plot showing the inventory over time (for each week) up until the inventory becomes zero.

Algorithm Requirements:

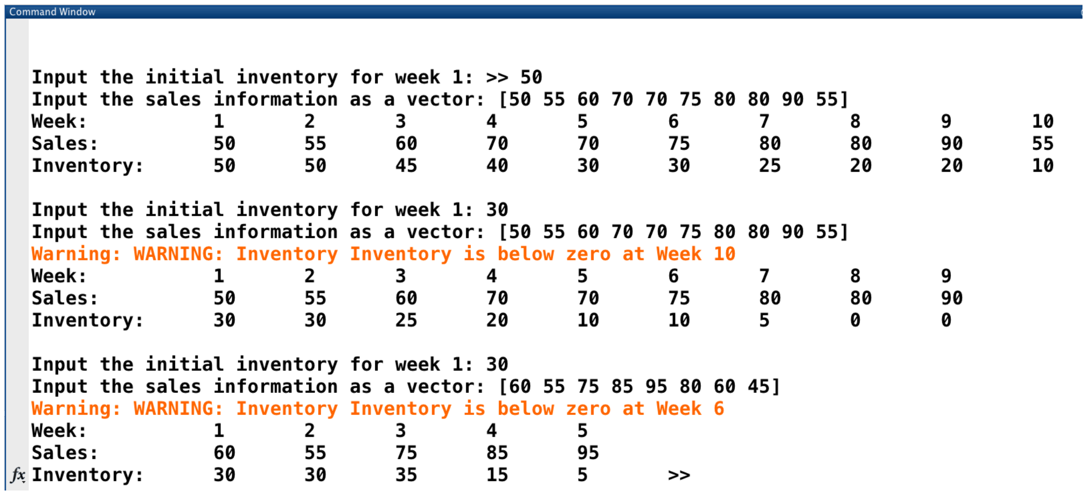
- Create an updated algorithm with the modifications described above. It can be typed or by hand.
- Save or scan the file to a .pdf file and include in your final .zip submission

Coding Requirements:

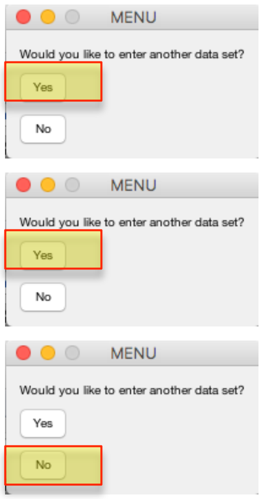
- Data validation (While Loops): If the initial inventory is less than or equal to zero, ask the user to enter a new initial inventory until then have entered an appropriate value.
- Data validation (While Loops): If any value is the vector of sales is less than zero, warn the user and ask the use to enter a new value. Before moving on to check the next value, keep asking until the user has entered an appropriate value.
- Use For loops for data processing and calculations
- User input for initial inventory
- User input for weekly sales
- User menu for repeat of program
- Formatted output for Summary Table shown on command window

Sample Output to Command Window On Next Page

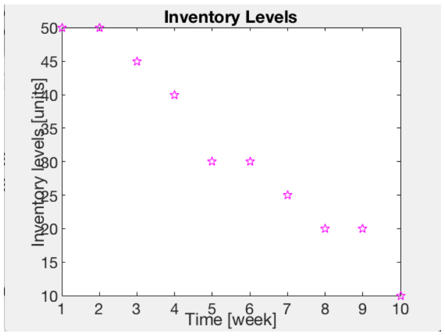
Sample Output to Command Window

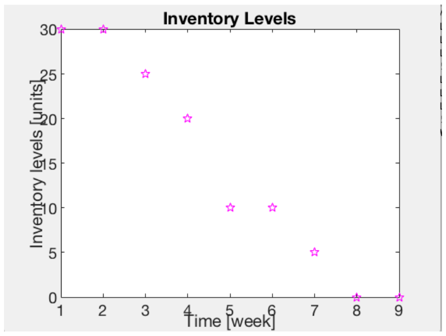


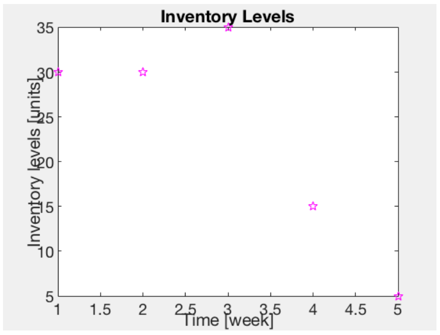
```
Command Window

Input the initial inventory for week 1: >> 50
Input the sales information as a vector: [50 55 60 70 70 75 80 80 90 55]
Week:           1       2       3       4       5       6       7       8       9       10
Sales:          50      55      60      70      70      75      80      80      90      55
Inventory:      50      50      45      40      30      30      25      20      20      10

Input the initial inventory for week 1: 30
Input the sales information as a vector: [50 55 60 70 70 75 80 80 90 55]
Warning: WARNING: Inventory Inventory is below zero at Week 10
Week:           1       2       3       4       5       6       7       8       9
Sales:          50      55      60      70      70      75      80      80      90
Inventory:      30      30      25      20      10      10      5       0       0

Input the initial inventory for week 1: 30
Input the sales information as a vector: [60 55 75 85 95 80 60 45]
Warning: WARNING: Inventory Inventory is below zero at Week 6
Week:           1       2       3       4       5
Sales:          60      55      75      85      95
fx Inventory:   30      30      35      15      5       >>
```


MENU
Would you like to enter another data set?
Yes
No
MENU
Would you like to enter another data set?
Yes
No
MENU
Would you like to enter another data set?
Yes
No

Plot for Entry #1: Plot for Entry #2 Plot for Entry #3


Inventory Levels
Inventory levels [units]
Time [week]


Inventory Levels
Inventory levels [units]
Time [week]


Inventory Levels
Inventory levels [units]
Time [week]