General Instructions

Due Date:
Saturday, March 11 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: Flipper Program
- Problem #2: Maximum and Minimum Program
- Problem #3: Problem Solving and Application to Solar Panels

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, MA6_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, MA6_dwburles.zip).** The file must be completely submit before 11pm on March 11 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: Write your own "flipper" script where the user inputs a matrix of any size. This script simply takes a given matrix and flips it from left to right, each row in this matrix.

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 nested For Loops
- Do **NOT** use the built-in function fliplr( ) or sort( ) or similar functions.
- Also, implicit Loops are **NOT** allowed. Do not use the colon ":" operator to address array elements such as A(:,k) or A(g,:).
- You **MUST** use explicit loops (For loops)
- User input for matrix
- Formatted output as shown in the Sample Output below.

Sample Output (Test Cases) to Command Window

Command Window

```
Enter a Matrix of your choosing: >> [1,2,3,4;5,6,7,8;9,10,11,12]

Your new Matrix is:
     4     3     2     1
     8     7     6     5
    12    11    10     9
```

Command Window

```
Enter a Matrix of your choosing: >> [1,2;3,4;5,6;7,8;9,10]

Your new Matrix is:
     2     1
     4     3
     6     5
     8     7
    10     9
```

Command Window

```
Enter a Matrix of your choosing: >> [10,9,8;7,6,5;4,3,2;1,0,-1]

Your new Matrix is:
     8     9    10
     5     6     7
     2     3     4
    -1     0     1
```

Problem #2

Background: Write a script that determines the overall maximum and minimum value of a matrix and the location of each **WITHOUT** using the max or min function or implicit loops. Allow the user to enter a matrix

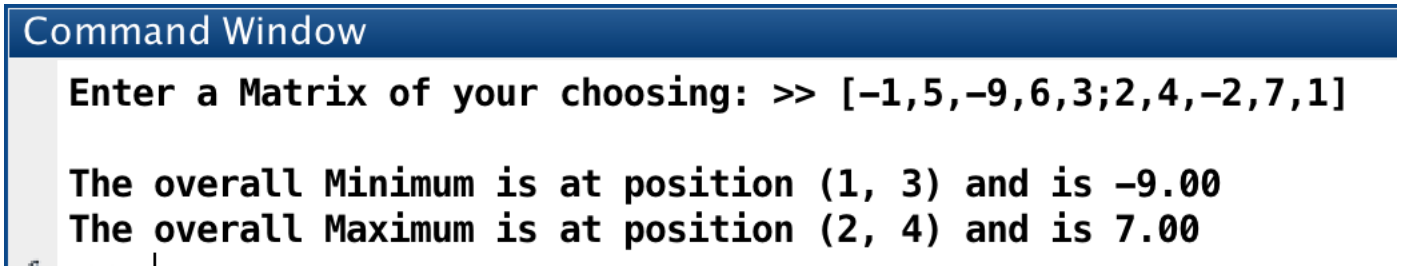
Algorithm Requirements:

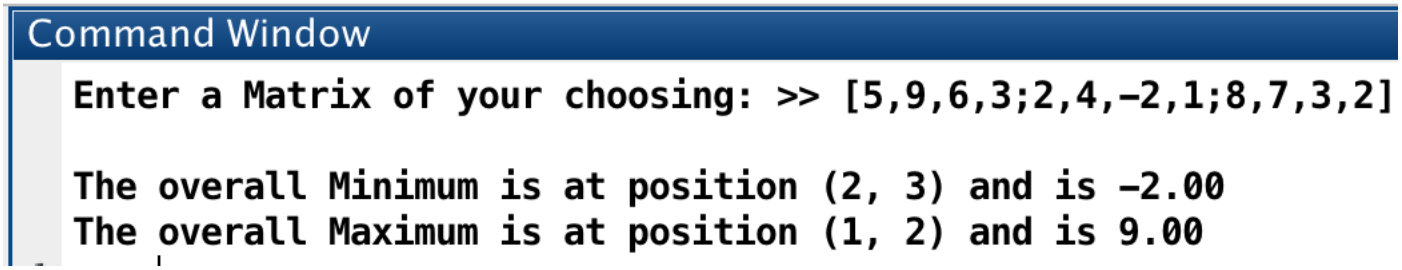
- Create an updated algorithm with the modifications described above. It can be typed or by hand.
- Save or scan the file to a .pdf file and include in your final .zip submission

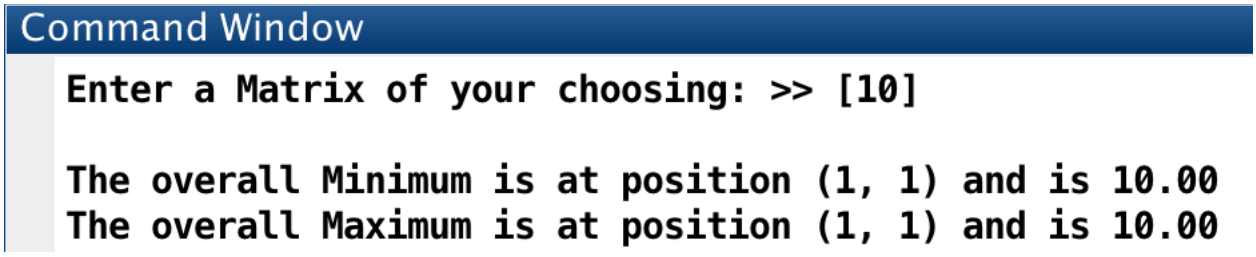
Coding Requirements:

- User input for the matrix
- Do not use the built-in function max or min or any similar function.
- Implicit Loops are **NOT** allowed. Do not use the colon ":" operator to address array elements such as A(:,k) or A(g,:).
- You **MUST** use explicit loops (For loops)

Sample Output (Test Cases) to Command Window







Problem #3

Background: *In this problem, assume that you have designed a large array of small photovoltaic cells, which are very cheap but fail often. You need to design a system which can monitor the current received irradiation data and alert if any cells are reporting outside of the range of their surrounding neighbors.*

The data file provided Problem3.csv. The file will be a matrix of many rows and columns. Each element represents a single PV cell that you will be checking for failures.

First, your code should check that all values in the matrix are less than 1000. For each value found that is greater than 1000, ask the user to replace that value with a new value less than 1000. If a value less than 1000 is not entered, the program should ask again until a value less than 1000 is entered. The result should be a matrix with all values less than 1000. This must be checked before calculations begin.

You will search for any failed PV cells by averaging the value of all cells around the cell of interest and using the percent difference given in the equation below to compare that average with the value of the cell of interest. The user will enter a percent difference threshold which will determine at what level a cell is considered failing.

Your code should perform these checks and calculations at each cell keeping in mind the multiple cases described at the bottom of this page with associated diagrams:

- *Average value of all cells around the center cell.*
- *Use the percent difference equation below with "**a**" as the average value "**b**" as the center cell value.*

$$Percent\ Difference = \left|\frac{a-b}{(a+b)/2}\right| * 100\%$$

The percent difference value will be used to determine if a cell is failing. If the percent difference found for each target cell is greater than the user inputted percent difference, store the row, column, value, average around the cell, and percent difference in a single matrix that contains only the failed cells. You must store each cell that fails in this matrix without overwriting the previous value. HINT: the number of rows in your failed cells matrix should equal the number of cells that fail based on the user input percent difference.

711	669	703
694	654	785
786	766	652

Cell in the middle (8 surrounding cells)

832	861
741	826
799	856

Cell corner (3 surrounding cells)

711	669	703
694	654	785
786	766	652

Cell on sides (5 surrounding cells)

711	669	703	832	861
694	654	785	741	826
786	766	652	799	856

Cell on top and bottom

Your formatted output table should include the position of each failing cell (Row and Column) along with the PV cell value for that location, the average of the PV cell values surrounding the cell of interest, and the calculated percent difference. The

*format should look like the sample output provided on the next page. You **MUST** use a loop to produce this output or have each line output within an existing loop*

[EXTRA CREDIT] Figure out a way to visualize the readings from the PV cells. Make sure there are appropriate labels and includes all necessary elements required for a plot.

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:

- For loops for formatted output
- User input for percent difference threshold
- NOTE: The grading data will include an array of values that is a different size than the test data provided. Therefore, your code must be flexible to handle a different number of rows and columns.

Test Case Sample Output

1. Load file *Problem3.csv*
2. Replace the following values when requested:
   - Cell (1,50) replace with 1200
   - Cell (1,50) asked again and replace with 900
   - Cell (5,9) replace with 200
   - Cell (25,14) replace with 3000
   - Cell (25,14) replace with 2000
   - Cell (25,14) replace with 600
3. Set Percent Difference check at 50

Command Window

```
Please replace Cell (1,50) with a new value:   >> 1200
Please replace Cell (1,50) with a new value:   900
Please replace Cell (5,9) with a new value:   200
Please replace Cell (25,14) with a new value:   3000
Please replace Cell (25,14) with a new value:   2000
Please replace Cell (25,14) with a new value:   600
Enter the Percentage Differance check for failed cells: 50

 Row#   Col#    PV Cell Avg around PV   Perrcent Difference
 5      9       200     749.88          115.78
 21     33      205     808.62          119.10
 46     10      360     763.00          71.77
 62     45      42      730.12          178.24
 67     1       153     774.40          134.01
 135    34      0       782.88          200.00
 170    10      405     745.50          59.19
 171    10      210     738.25          111.42
 194    41      162     756.12          129.42
 200    50      96      463.20          131.33
fx >>
```