# Week 2 Homework Assignments: MATLAB Fundamentals

## Global Requirements

- All deliverables shall be added, committed, and pushed to your `Week2` folder in your repository.
- All files shall have your name and the name of anyone who you worked with using the following format.

```
% Student: Firstname Lastname
% Assisted by: first person, second person, etc.
```

---

## 1. Matrix Arithmetic

### Instructions

Write a MATLAB script that performs basic matrix and vector arithmetic.

**Script Name**: `matrixOperations.m`

**Objectives**:

- Take note of the results, see how the element-wise operations are applied.

**Steps**:

1. **Initialize a Matrix**:
   - Create a 3x3 matrix `A` where each element is a number between 1 and 9:
     `A = [1, 2, 3; 4, 5, 6; 7, 8, 9];`
   - Create another 3x3 matrix `B` where each element is a number between 9 and 1:
     `B = [9, 8, 7; 6, 5, 4; 3, 2, 1];`
2. **Matrix Addition**:
   - Add matrices `A` and `B` and store the result in a new matrix `C`.
3. **Matrix Subtraction**:
   - Subtract matrix `B` from matrix `A` and store the result in a new matrix `D`.
4. **Element-wise Multiplication**:
   - Multiply matrices `A` and `B` element by element (use the `.*` operator) and store the result in matrix `E`.
5. **Matrix Scalar Multiplication**:
   - Multiply matrix `A` by a scalar value (for example, 2) and store the result in matrix `F`.

### Deliverables

1. Submit the script file (`matrixOperations.m`) that contains all matrix operations and their results.
2. Include comments explaining what each part of the code does.

---

## 2: Vector Arithmetic

### Instructions

Write a MATLAB script that demonstrates basic vector operations.

**Script Name**: `vectorOperations.m`

**Steps**:

1. **Create a Row Vector**:
   - Define a row vector `x` with elements [1, 3, 5, 7, 9].
2. **Create a Column Vector**:

- Define a column vector `y` with elements [2; 4; 6; 8; 10].

3. **Element-wise Multiplication**:
   - Multiply the row vector `x` by the column vector `y` element by element (use the `.*` operator).
4. **Vector Transposition**:
   - Transpose the row vector `x` to a column vector.
5. **Vector Summation**:
   - Find the sum of all elements in vector `x` using the `sum` function.

**Deliverables**

1. Submit the script file (`vectorOperations.m`) that includes all vector operations.
2. Include comments explaining what each part of the code does.

---

## 3: Simple Data Visualization

**Instructions**

Write a MATLAB script that creates a basic line plot.

**Script Name**: `simplePlot.m`

**Steps**:

1. **Create a Time Vector**:
   - Define a vector `time = [0, 1, 2, 3, 4, 5];`.
2. **Create a Distance Vector**:
   - Define a vector `distance = [0, 10, 20, 30, 40, 50];`.
3. **Plot the Data**:
   - Use the `plot` function to plot `time` on the x-axis and `distance` on the y-axis.
4. **Add Labels and Title**:
   - Label the x-axis as "Time (s)" and the y-axis as "Distance (m)".
   - Add a title to the plot: "Time vs Distance".

**Deliverables**

1. Submit the script file (`simplePlot.m`) that creates the plot.
2. Include comments explaining what each part of the code does.

---

## 4. Comprehensive MATLAB Script

**Task**

Create a MATLAB script that performs multiple calculations related to material properties and visualizes the results.

**Instructions**

**Script Name**: `materialProperties.m`

**Objectives**:

- Calculate stress and strain for a material using vectors.
- Visualize the stress-strain relationship with a plot.

**Steps**:

1. **Variable Initialization**:

- Define a vector `forces = [100, 200, 300, 400, 500];` (in Newtons).
- Define a scalar `crossSectionArea = 50;` (in square millimeters).
- Define a vector `displacements = [0.1, 0.2, 0.3, 0.4, 0.5];` (in millimeters).
- Define a scalar `originalLength = 100;` (in millimeters).

2. **Stress Calculation**:
   - Calculate the stress using the formula
   $$\text{stress} = \frac{\text{force}}{\text{crossSectionArea}}$$
   for each force value.
   - Store the results in a vector `stress`.
3. **Strain Calculation**:
   - Calculate the strain using
   $$\text{strain} = \frac{\text{displacement}}{\text{originalLength}}$$
   for each displacement value.
   - Store the results in a vector `strain`.
4. **Stress-Strain Plot**:
   - Plot the stress-strain curve using the calculated values.
   - Label the x-axis as "Strain" and the y-axis as "Stress (MPa)".
   - Add a title "Stress-Strain Curve".

**Deliverables**

1. Submit the script file (`materialProperties.m`) containing all calculations and the plot.
2. Include comments explaining each calculation and the purpose of the plot.

---

## 5. Comprehensive MATLAB Script: Physics Simulation and Analysis Toolbox

**Task**

Simulate the motion of a projectile under gravity and visualize the results.

**Instructions**

**Script Name**: `projectileMotion.m`

**Objectives**:

- Simulate vertical motion under gravity using given initial conditions.
- Visualize the motion using a time vs. height plot.

**Steps**:

1. **Variable Initialization**:
   - Define constants: `gravity = 9.81;` and `initialVelocity = 50;` (in m/s).
   - Create a time vector `time = 0:0.1:10;` (representing 0 to 10 seconds).
2. **Height Calculation**:
   - Calculate the height of the projectile at each time point using
   $$y = \text{initialVelocity} \times \text{time} - \frac{1}{2} \times \text{gravity} \times \text{time}^2$$
   - Store the results in a vector `height`.
3. **Plotting the Trajectory**:
   - Use `plot(time, height);` to create the plot.
   - Label the x-axis as "Time (s)" and the y-axis as "Height (m)".
   - Add a title "Projectile Motion Under Gravity".

### Deliverables

1. Submit the script file (`projectileMotion.m`) with all calculations and the plot.
2. Include comments explaining each calculation and the purpose of the plot.

---

## 6. Bug Hunt Challenge

### Task

Identify and fix errors in a MATLAB script that calculates the total cost of items in a grocery list and displays the result.

### Instructions

1. Copy the provided buggy MATLAB code to a script file named `buggyScript2.m`.
2. Open the script in the MATLAB Editor and try running it. Observe the errors or unexpected behaviors.
3. Identify and fix the bugs in the script. The bugs could include syntax errors, incorrect operations, or function misuse.
   - Pay attention to how arrays and variables are used and manipulated.
4. Use comments to explain each fix you make and describe the original error.

### Example Buggy Script (buggyScript2.m)

```
% Task: Calculate the total cost of items in a grocery list

itemPrices = [2.5, 3.0, 4.5, 5]; % Prices of 4 items
itemQuantities = [2, 1, 3]; % Quantities of each item bought

% Calculate the total cost for each item
totalCost = itemPrices * itemQuantities;

% Display the total cost (using sum to get the overall cost)
fprintf('The total cost of the grocery items is: %.2f\n', sum(totalCost));
```

### Deliverables

1. Submit the corrected script file (fixedScript2.m).
   - Ensure that all calculations and logic work as intended.
   - Include comments explaining each error you found and how you fixed it.
2. Write a short report (debuggingReport2.txt) containing the following:
   - Summarize the errors you encountered.
   - Explain how you found the solution to fix them.
   - Explain what you learned from the debugging process.

---

## Definition of Done

1. You shall have a GitHub Repository set up with gberl001 invited as a collaborator.
2. Your Week2 Folder shall have the following files:
   - matrixOperations.m
   - vectorOperations.m
   - simplePlot.m
   - materialProperties.m

- projectileMotion.m
- buggyScript2.m
- fixedScript2.m
- debuggingReport2.txt