Week 5 Assignments: Logical Indexing & Matrix Operations

Global Requirements

- All deliverables shall be added, committed, and pushed to your `Week5` folder in your repository.
- Include your name and the names of anyone who assisted you in the following format at the top of each `.m` file:

```
% Student: Firstname Lastname
% Assisted by: Firstname Lastname, etc.
```

- Ensure your scripts (not functions) include the following to clear the workspace and command window:

```
clc;
clear;
```

- Any CSV files should be read from or written to in the same directory as your scripts by default. (No file path shenanigans)

1. Material Strength Filter Tool

> [!CAUTION] No loops allowed in this assignment. Remember to review MATLAB's documentation

Task

Create a MATLAB script and function that help engineers filter materials based on their mechanical properties using logical vectors and operations.

Background Engineers often need to select materials that meet specific criteria for strength, weight, and cost. This tool will assist in filtering materials from a dataset based on user-defined thresholds.

Dataset You are provided with a CSV file named `material_properties.csv`, which contains the following columns:

- `Material`: Name of the material.
- `Density`: Density in kg/m³.
- `TensileStrength`: Tensile strength in MPa.
- `CostPerKg`: Cost per kilogram in USD.

Example `material_properties.csv`:

```
Material,Density,TensileStrength,CostPerKg
Aluminum,2700,300,1.5
Steel,7850,500,0.8
Titanium,4500,900,12
CarbonFiber,1600,3500,20
Plastic,950,50,0.5
```

Function: filterMaterials

Requirements:

- **Inputs**:
  - `filename`: Name of the CSV file (string).
  - `maxDensity`: Maximum acceptable density (scalar).
  - `minTensileStrength`: Minimum required tensile strength (scalar).
  - `maxCost`: Maximum acceptable cost per kg (scalar).
- **Outputs**:

- `filteredMaterials`: A table containing materials that meet the criteria.

Instructions:

Write a MATLAB function named `filterMaterials.m`. * Read the material data from the CSV file into a table. * Use logical vectors to filter materials based on the following conditions: * Density <= maxDensity * TensileStrength >= minTensileStrength * CostPerKg <= maxCost * Return the filtered list of materials.

Script: `materialSelector.m`

Requirements:

1. **User Inputs**:
   - Prompt the user to input the maximum acceptable density.
   - Prompt the user to input the minimum required tensile strength.
   - Prompt the user to input the maximum acceptable cost per kg.
2. **Manipulation**:
   - Call the `filterMaterials` function with the user inputs.
3. **Output**:
   - Display the filtered list of materials.
   - If no materials meet the criteria, display an appropriate message.

Example Interaction:

```
Enter the maximum acceptable density (kg/m^3): 5000
Enter the minimum required tensile strength (MPa): 400
Enter the maximum acceptable cost per kg (USD): 15
```

Example Output:

```
Materials that meet your criteria:
  Material      Density    TensileStrength    CostPerKg
 ----------     -------    ---------------    ---------
{'Titanium'}      4500           900              12
```

Testing

- Test your function with different input values to ensure it works correctly.
- Ensure that your script handles cases where no materials meet the criteria.
- `testFilterMaterials.m` tests the `filterMaterials()` function

Deliverables

- Submit the function `filterMaterials.m`.
- Submit the script `materialSelector.m`.
- Include the `material_properties.csv` file in your `Week5` folder.
- Ensure your code is well-commented and uses logical vectors for filtering.

2. Smart Inventory Alert System

[!CAUTION] No loops allowed in this assignment. Remember to review MATLAB's documentation

Task

Enhance your kitchen inventory management system to include an automatic alert for low-stock items using logical vectors and operations.

Background In a smart kitchen, it's essential to keep track of ingredient quantities and receive alerts when stocks are low to ensure seamless meal preparation.

Function: `getLowStockItems`

Requirements:

- **Inputs**:
  - `filename`: Name of the inventory CSV file (e.g., `inventory.csv`). Copy your inventory file from the previous week if necessary
  - `threshold`: Quantity threshold for low stock (scalar).
- **Outputs**:
  - `lowStockItems`: A table containing items with quantities less than or equal to the threshold.

Instructions:

- Write a MATLAB function named `getLowStockItems.m`.
- Read the inventory data from the CSV file into a table.
- Use logical vectors to identify items where qty <= threshold.
- Return a table of low-stock items.

Script: `kitchenInventory.m` (Updated)

Additional Menu Option:

- Add a new option to the main menu:
  - "4. Check for low-stock items"

Functionality:

- When the user selects option 4:
  - Prompt the user to enter the low-stock threshold.
  - Call getLowStockItems with the user-specified threshold.
  - Display the list of low-stock items.
  - If no items are low on stock, display an appropriate message.

Example Interaction:

```
    ----------------------------------
| Welcome to the Kitchen Inventory Manager!
|
| Please select an option:
| 1. Add an ingredient
| 2. Print inventory list
| 3. Check ingredient quantity by UPC
| 4. Check for low-stock items
| 0. Exit
> 1
Enter the UPC: 123654
Enter the ingredient name: Wheat Bread
Enter the quantity: 1
Ingredient added successfully.

    ----------------------------------
| Welcome to the Kitchen Inventory Manager!
|
| Please select an option:
| 1. Add an ingredient
```

```
| 2. Print inventory list
| 3. Check ingredient quantity by UPC
| 4. Check for low-stock items
| 0. Exit
> 4
Enter the low-stock threshold: 2
Items low on stock:
       upc          ingredient       qty
    ----------    ---------------    ---

    1.2365e+05    {'Wheat Bread'}     1

    ----------------------------------
| Welcome to the Kitchen Inventory Manager!
|
| Please select an option:
| 1. Add an ingredient
| 2. Print inventory list
| 3. Check ingredient quantity by UPC
| 4. Check for low-stock items
| 0. Exit
```

Testing

- Test the updated script by adding ingredients with varying quantities.
- Verify that the getLowStockItems function correctly identifies low-stock items.
- `testGetLowStockItems.m` tests the `getLowStockItems()` function by verifying the table output

Deliverables

1. Submit the updated function `getLowStockItems.m`.
2. Submit the updated script `kitchenInventory.m`.
3. Ensure your code uses logical vectors for identifying low-stock items.
4. Include comments explaining the new functionality.

3. Data Analysis of Projectile Trajectories

Task

Analyze projectile trajectory data to identify key characteristics using logical vectors and operations.

Background Physicists often need to analyze simulation data to extract meaningful insights. This assignment focuses on analyzing the projectile trajectories generated last week.

Function: `analyzeTrajectories`

Requirements:

- **Inputs**:
  - `angles`: Vector of launch angles in degrees.
  - `ranges`: Vector of corresponding projectile ranges in meters.
- **Outputs**:
  - `aboveAverageAngles`: Vector of angles that resulted in above-average ranges.
  - `maxRange`: The maximum range achieved (scalar).
  - `optimalAngle`: The angle corresponding to the maximum range (scalar).

Instructions:

- Write a MATLAB function named `analyzeTrajectories.m`.
- Calculate the average range.
- Use logical vectors to find angles where the range is above average.
- Identify the maximum range and the corresponding angle.

Script: `trajectoryAnalysis.m`

Requirements:

1. **Data Generation**:
   - Generate a set of projectile ranges for angles from 0 to 90 degrees using last week's `calculateTrajectory()` function.
2. **Data Analysis**:
   - Use analyzeTrajectories to analyze the data. .
3. **Output**:
   - Display the average range, maximum range, and optimal angle in the command window.
   - Display the angles that resulted in above-average ranges.

Example Output:

```
Average Range: 45.00 meters
Maximum Range: 60.00 meters at an angle of 45 degrees
Angles resulting in above-average ranges:
[40 41 42 43 44 45 46 47 48 49 50]

Average Range: 70.47 meters
Maximum Range: 101.25 meters at an angle of 42 degrees
Angles resulting in above-average ranges:
  Columns 1 through 18

    16    17    18    19    20    21    22    23    24    25    26    27    28    29    30    31    32

  Columns 19 through 36

    34    35    36    37    38    39    40    41    42    43    44    45    46    47    48    49    50

  Columns 37 through 51

    52    53    54    55    56    57    58    59    60    61    62    63    64    65    66
```

Testing

- Verify that your script correctly identifies the optimal angle and above-average angles.
- Test with different initial velocities and heights to see how the results change.

Deliverables

1. Submit the function analyzeTrajectories.m.
2. Submit the script trajectoryAnalysis.m.
3. Include any necessary functions from last week, if modified.
4. Ensure your code is well-commented and uses logical vectors for analysis.

Definition of Done

Your `Week5` folder shall contain at minimum the following files:

- analyzeTrajectories.m
- filterMaterials.m
- getLowStockItems.m
- kitchenInventory.m (updated)
- materialSelector.m
- material_properties.csv
- trajectoryAnalysis.m
- Any other functions or scripts you created or modified

Additional Instructions and Tips

- Logical Vectors and Operations:
  - Use relational operators (<, >, <=, >=, ==, ~=) to create logical vectors.
  - Utilize logical indexing to filter data efficiently.
- Data Handling:
  - Use MATLAB's table data type for handling CSV data (readtable, writetable).
  - Ensure your programs can handle any number of data entries.
- Plotting:
  - Label your axes and provide titles for your plots.
  - Use legends to distinguish different data series.
- User Interaction:
  - Validate user inputs where appropriate.
  - Provide clear prompts and messages.
- Code Quality:
  - Include comments explaining your logic and code sections.
  - Use meaningful variable names.