MA #2 Guide

Overall Goal: You will practice your knowledge of vectors, matrices, and functions to evaluate data for an ice patch. Ice thickness measurements have been taken on various days for different locations and are stored in the variable **Ice**.

What is a .p file?

Included in the MA #2 files you will see a .p file. This is a locked file (you cannot open or edit it) that contains the correct function code. Place this file in your current folder. Then, you can call this function in your main script the same way you would call any other function.

Why would you use the .p file?

If your code is not matching the sample output and you are not sure why, use the .p file. If you run your code and call the p.file function and you get the correct sample output, then you have identified that the error is in your function file and not the main script. For the MA, you get separate credit for a correct main script and a correct function.

MA Requirements:

Algorithm: Submitted to Blackboard
MATLAB Code: Submitted to Zybooks

Task 1: Main Script – Load data and analyze based on user inputs

- Load the data in the file: **MA2_data.mat** and familiarize yourself with the data contained in the **Days**, **Ice,** and **LocationID** variables
- Using a menu
  - Prompt the user to select a day.
  - HINT: The options for a menu ***always*** have to be string arrays. Numbers must be converted from math numbers to text numbers to be used in a menu. The **string** function could be useful here! Type `help string` in the MATLAB command window to explore how the string function works!
  - Prompt the user to select a location
- Based on the user selections output to the command window the day, location, and corresponding ice thickness.

Task 2: Main Script – Add new data that was collected

- Prompt the user to input a vector that contains 5 ice thickness values, one for each of the locations in **Ice**
- Add the new ice thickness measurements to the bottom of **Ice**
- The new measurements were taken 5 days after the last entry. Add the new day number to **Days**
  - HINT: Avoid hardcoding. Typing 60 + 5 is hardcoding because you are manually typing the value stored in the last entry in **Days** instead of using variables and indexing.
- A new location was also discovered 5 days after the last entry:
  - Prompt the user to enter the name of a new location that was discovered.
  - Prompt the user to enter one ice measurement that corresponds to measurement on the discovery day.
- Because you only have data for the discovery day (user-entry), assume the ice measurements on the days in the original **Ice** matrix for the new location are the daily averages of the measurements from the five original locations.
- Update the **Ice** matrix with the measurements for the new location in the last column.

- Update the **LocationID** variable with the user-entered location as the last entry.
- Save the updated **Ice**, **LocationID**, and **Days** variables as **MA2_Task2.mat**
- Output to the command window the name of the new location ID **and** the average ice thickness on the last day in **Ice.**

Task 3: Main Script – Perform summary statistics on the updated data in Ice from Task 2

- Output to the command window the number of locations **and** the total number measurements
- Output to the command window the location with highest average thickness **and** the corresponding value
- Output to the command window the overall maximum ice measurement (from all the data), the corresponding day, and the corresponding location
- Output to the command window the overall average ice thickness (from all the data)
  - HINT: Avoid hardcoding: You should be using variables to index from **Ice** and **LocationID**. Your code should still work if you were suddenly presented with a new data set where the maximum values corresponded to different days or locations.

Task 4: Main Script – Predict the ice thickness on a future day

- Prompt the user to input as a percentage how much the ice thickness has reduced. The entered value should be a number between 0 – 100.
- Call your Projection.m function to predict the ice thickness at all the locations from Task 2 on a future date.
  - Inputs: (1) Percent ice reduction from the user input, (2) **Ice** matrix from Task 2
  - Outputs (1) Vector storing predicted ice thicknesses for each location
- Add the predicted ice thickness values to the **Ice** matrix from Task 2
- Export the updated **Ice** matrix as a .csv file named **MA2_Task4.csv**

Task 4: Function – Predict the ice thickness on a future day

- Create a new function named **Projection.m**
  - Inputs: (1) Percent ice reduction from the user input, (2) **Ice** matrix from Task 2
  - Outputs (1) Vector storing predicted ice thicknesses for each location
- Within your function, calculate the predicted ice thickness for the future day. Your calculation should assume the ice thickness for the last day in **Ice** reduces by the percentage specified by the user.
  - HINT: Don't forget to change the numerical percentage value to a decimal!

Task 5: Main Script – Plot the ice thickness for an individual location

- Using a menu, prompt the user to select a location from the **LocationID** variable
- Create a plot to show the ice thickness from that location on all days including the predicted thickness from Task 4. Your plot must include: title, axis labels, gridlines, and data as individual points
- Plot 1: **x-axis:** Days; **y-axis:** Ice Thickness
  - BONUS: Explore the **sprintf** function to incorporate the location name in the title and have it update based on the user-selection!