

MATLAB Take Home Exam 02

CEE/GLE 291 Spring 2021

Instructions

Complete the following problems as part your exam. Submit the code for each function via Canvas. When submitting the code for a function, make sure the file is named accordingly. For example, when submitting the code for **getRotatedPoint**, the file containing the code must be named **getRotatedPoint.m** and must remain that regardless of you completing a new submission. If you overwrite a previous your submission in Canvas, make sure you overwrite your complete submission (i.e., submit all files every time you complete a submission even if you only are trying to update one file); otherwise, Canvas will show an incomplete submission as the last one.

Testing for Boundary Conditions

As part of the exam, you are provided some examples of function calls and along with the values returned by the sample function calls. However, you are responsible for making sure your functions work under different boundary conditions that result from the range of input values possible as well as addressing the full instructions in the problem statement. The identification of boundary conditions and correctly structuring the code to handle such scenarios is a key component of the exam. Therefore, do not assume that all possible input and expected value combinations are shown, e.g., do not hardcode behaviors, your code must provide the general solution.

Problem 01 [3 pts]

The function receives the x-y coordinates **coords1** and **coords2** that can be used to define two lines. The information that defines each line is passed as input in [x1, y1, x2, y2] form. The function returns the two angles defined by the intersection of the two lines in [smallest, largest] form. If the lines do not intersect, then the function returns [0, 0].

Function Name and Parameters

`intersectionAngle(coords1, coords2)`

Sample Function Call	Returned
<code>intersectionAngle([5,6,9,4], [4,5,8,6])</code>	<code>[40.6013, 139.3987]</code>
<code>intersectionAngle([6,7,8,3], [4,5,8,6])</code>	<code>[77.4712, 102.5288]</code>

Problem 02 [2 pts]

The function receives a character array **rowOfData** and a number **col** as input. The character array **rowOfData** represents a row of data in a comma-separated values (CSV) format. The number **col** represents the column of interest within the row of data. The function returns the value (as a character array or double) associated with the column. Therefore, if the value can be represented as a double then a double must be returned; if not, a character array must be returned. If an invalid column number is passed, the function returns the MATLAB NaN value.

Function Name and Parameters

`getValueAtColumn(rowOfData, col)`

Sample Function Call	Returned
<code>getValueAtColumn('a,b,c', 2)</code>	<code>'b'</code>
<code>getValueAtColumn('a,b,34', 3)</code>	<code>[34.0000]</code>
<code>getValueAtColumn('20.52,ced', 1)</code>	<code>[20.5200]</code>
<code>getValueAtColumn('a,b,34', 4)</code>	<code>NaN</code>

Problem 03 [2 pts]

The function receives character array **cArr** as input. The function sequentially evaluates every character in the array and returns the number of times there is a change in the case of the **cArr** characters. A non a-z/A-Z character should be treated as having the case of the last a-z/A-Z character (in left to right order).

Function Name and Parameters

```
caseChangeCount(cArr)
```

Sample Function Call	Returned
caseChangeCount('aBcrela')	[2]
caseChangeCount('rTREmnM')	[3]
caseChangeCount('aaaerw')	[0]

Problem 04 [2 pts]

The function receives a character array **cArr** as input. The function returns a sorted version of the array (in ascending order) in which the first part of the array contains the sorted version of the lowercase characters while the second part contains the sorted version of the uppercase characters. You can assume that the character array passed as input contains only a-z or A-Z characters. Please note that only unique characters are included in the value returned.

Function Name and Parameters

```
sortSplitCase(cArr)
```

Sample Function Call	Returned
sortSplitCase('aaDremE')	'aemrDE'
sortSplitCase('avttea')	'aetv'
sortSplitCase('PMEDDC')	'CDEMP'

Problem 05 [2 pts]

The function receives a character array **nameStr** containing a full name in uppercase, lowercase, or a mixed format as input. The function returns a character array file with the full named passed as input but formatted using sentence case. For simplicity purposes, you can assume that names only contain a-z or A-Z characters.

Function Name and Parameters

```
fixNameFormat(nameStr)
```

Sample Function Call	Returned
fixNameFormat('John SMITH')	'John Smith'
fixNameFormat('JANE DOE')	'Jane Doe'
fixNameFormat('JoHn SMITH')	'John Smith'
fixNameFormat('JANe DOe')	'Jane Doe'
fixNameFormat('')	''

Problem 06 [3 pts]

The function receives a character array **myStr** as input. The function returns a logical value indicating if the character array passed is symmetrical (i.e. a palindrome) or not. An empty character array passed as input must be treated as symmetrical. The presence of a non A-Z or a-z character results in the function returning an empty array. *Notice that logical(1) and logical(0) are simply used to indicate that a logical value is returned; it does not mean that the function returns a character array that reads like 'logical(1)'. The function is case-sensitive.*

Function Name and Parameters

isSymmetrical(myStr)

Sample Function Call	Returned
isSymmetrical('aba')	logical(1)
isSymmetrical('abba')	logical(1)
isSymmetrical('aara')	logical(0)
isSymmetrical('a##a')	[]
isSymmetrical('')	logical(1)

Problem 07 [3 pts]

The function receives the name/path of an image file **imgPath** as input as well as an array **targetColor** in RGB format. The function returns the number of pixels along the entire edges of the image that are of the specified color. If an invalid (or non-existing) image path is passed as input then an empty array is returned. Notice that in the sample function calls the *InvalidImage.tif* and *NonImage.docx* file are meant to represent a non-existing file or an existing file that is not an image.

Function Name and Parameters

edgeCount(imgPath, targetColor)

Sample Function Call	Returned
edgeCount('Image12.tif', [0,0,0])	12
edgeCount('Image16.tif', [255,0,0])	21
edgeCount('Image17.tif', [0,255,0])	4
edgeCount('InvalidImage.tif', [255,255,0])	[]
edgeCount('NonImage.docx', [255,255,0])	[]

Problem 08 [4 pts]

The function receives character array **cArr** as input. The function returns the longest streak (of any character) in the character array passed as input. An empty character array causes the function to return 0 and an invalid input (such as a number) causes the function to return an empty array. Character comparison is case-sensitive.

Function Name and Parameters

longestStreak(cArr)

Sample Function Call	Returned
longestStreak('aaa\$\$\$\$tttPPppp')	[4]
longestStreak('')	[0]
longestStreak(31416)	[]

Problem 09 [2 pts]

The function receives the name/path of a photo **photoPath** as input. The function returns the result of multiplying the absolute values of the latitude and longitudes (in decimal format). If **photoPath** is invalid (or non-existing) an empty array is returned. If the photo is valid but does not contain latitude and longitude information, zero is returned. In the sample function calls, Hydrant99.jpg is meant to represent a non-existing file.

Function Name and Parameters

gpsDataMultiplication(photoPath)

Sample Function Call	Returned
gpsDataMultiplication('Hydrant01.jpg')	3852.7027
gpsDataMultiplication('Hydrant02.jpg')	3852.6989
gpsDataMultiplication('Hydrant99.jpg')	[]
gpsDataMultiplication('Image03.tif')	[0]

Problem 10 [2 pts]

The function receives as input a path to a MAT file **myData** containing a *Position (timetable)* dataset generated using the GPS logging capabilities of the MATLAB Mobile app as well as a distance threshold **dist** in meters. The function returns a logical value indicating if the vehicle path documented in the timetable started and ended at points within **dist** of each other. Use wgs84 as the hardcoded ellipsoid in your calculations. If **myData** is not a valid timetable, return an empty array.

Function Name and Parameters

segStats(myData, dist)

Sample Function Call	Returned
segStats('BusRide.mat', 100)	logical(1)
segStats('BusRide.mat', 5)	logical(0)
segStats('InvalidData.mat', 5)	[]