Problem Solving For Engineering Transfer

ENS 1300 - Spring 2020

Due Date	Lab 5: Loops	Last Modified
13 March 2020		9 March 2020

Problem 1

Create a function called `multTable` that accepts a single input (integer) and returns no output. The input will be the number of rows and columns to add to a multiplication table. Print the multiplication table to a text file called `multTable.txt` and separate each row value with a tab. The suggested approach is as follows:

1) Initialize an n by n matrix, where n is the input integer, using the `zeros` function.
2) Create an outer for loop to iterate over the rows of the matrix. Use `i` as the looping variable.
3) Create an inner (nested) for loop to iterate over the columns of the matrix. Use `j` as the looping variable.
4) Assign the value `i*j` to the matrix element `(i,j)`. Leave the definition unsupressed and step through the loops in debug mode. Watch the values of `i` and `j` change in the workspace and the redefinition of the matrix with each iteration.
5) Print the multiplication table to file as described in the paragraph above. (Note: storing the values in a matrix is not necessary, so you may get rid of it if you like)

Function Specifications:

Function Name	Purpose	Input(s)	Output(s)
`multTable`	write a multiplicaiton table to file	number of rows and columns for the table (integer)	none

Useful functions: `zeros`, `fopen`, `fprintf`

Deliverable(s): `multTable.m`

Graded Item	Point Value
Adequate comments and organization	5
Correct input/output	5
Total	10

Problem 2

Write a function called `piCalcs1` that accepts a single input (scalar double) and returns no output. The input will be a tolerance for calculating π using the following formulas:

Wallis

$$\frac{\pi}{2} = \prod_{n=1}^{\infty} \frac{(2n)^2}{(2n-1)(2n+1)}$$

Gregory and Leibniz

$$\frac{\pi}{4} = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1}$$

Bellard

$$\pi = \frac{1}{2^6} \sum_{n=0}^{\infty} \frac{(-1)^n}{2^{10n}} \left[-\frac{2^5}{4n+1} - \frac{1}{4n+3} + \frac{2^8}{10n+1} - \frac{2^6}{10n+3} - \frac{2^2}{10n+5} - \frac{2^2}{10n+7} + \frac{1}{10n+9} \right]$$

Borwein-Bailey

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left[\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right]$$

Euler

$$\pi = \sqrt{6 \sum_{k=1}^{\infty} \frac{1}{k^2}}$$

Each one of the formulas should be executed in their own subfunction. When the tolerance has been met, print out a message that includes the formula name, the tolerance, and the number of iterations performed. Add a conditional statement at the beginning of the main function to use a default tolerance of 10^{-4} if no input is provided.

Function Specifications:

Function Name	Purpose	Input(s)	Output(s)
`piCalcs1`	calculate π to within a specified tolerance using various formulas	tolerance value (scalar double)	none

Useful functions: `fprintf`

Deliverable(s): `piCalcs1.m`

Graded Item	Point Value
Adequate comments and organization	5
Correct input/output	15
Total	20

Problem 3

Write a function called `piCalcs2` that accepts a single input (integer) and returns no output. The input will be the number of iterations to perform on the following formulas for π:

Wallis

$$\frac{\pi}{2} = \prod_{n=1}^{\infty} \frac{(2n)^2}{(2n-1)(2n+1)}$$

Gregory and Leibniz

$$\frac{\pi}{4} = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1}$$

Bellard

$$\pi = \frac{1}{2^6} \sum_{n=0}^{\infty} \frac{(-1)^n}{2^{10n}} \left[-\frac{2^5}{4n+1} - \frac{1}{4n+3} + \frac{2^8}{10n+1} - \frac{2^6}{10n+3} - \frac{2^2}{10n+5} - \frac{2^2}{10n+7} + \frac{1}{10n+9} \right]$$

Borwein-Bailey

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left[\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right]$$

Euler

$$\pi = \sqrt{6 \sum_{k=1}^{\infty} \frac{1}{k^2}}$$

Each one of the formulas should be executed in their own subfunction. After the iterations are complete, create an appropriately annotated plot showing the convergence to π. Plot a horizontal line for π and use a line and markers for the formula iterations. Add a conditional statement at the beginning of the main function that will assign a default number of iterations if no input is provided.

Function Specifications:

Function Name	Purpose	Input(s)	Output(s)
`piCalcs2`	calculate π with a specified number of iterations using various formulas	number of iterations (integer)	none

Useful functions: `figure`, `plot`, `axis`, `xlabel`, `ylabel`, `legend`

Deliverable(s): `piCalcs2.m`

Graded Item	Point Value
Adequate comments and organization	5
Correct input/output	15
Total	20

Problem 4

Open a new MATLAB file (or start from our function template) and create a new function called `movingAverages` that accepts two inputs and returns no outputs. The first input will be a string array containing dates and the second will be a vector of daily stock or cryptocurrency price data. I have provided the last 10 years of daily Bitcoin prices on the course website in two different files. Manipulating the original CSV file downloaded from www.investing.com into a format that is useful for our purposes takes a quite a bit of work, so I have extracted everything you need and saved it in the `BTC.mat` file. The CSV file is included just in case you want to see it.

For this problem, you will plot the daily price data and the 5, 10, and 30 day moving averages for the last n days starting with February 23, 2020. For example, the 5-day average will be calculated by taking the mean of the current day and the previous 4 days. So if you wanted to plot the last 365 days, you will need to pass the last 394 days of data to the `movingAverages` function because the previous 29 days are needed to calculate each 30-day average. My suggested steps are as follows:

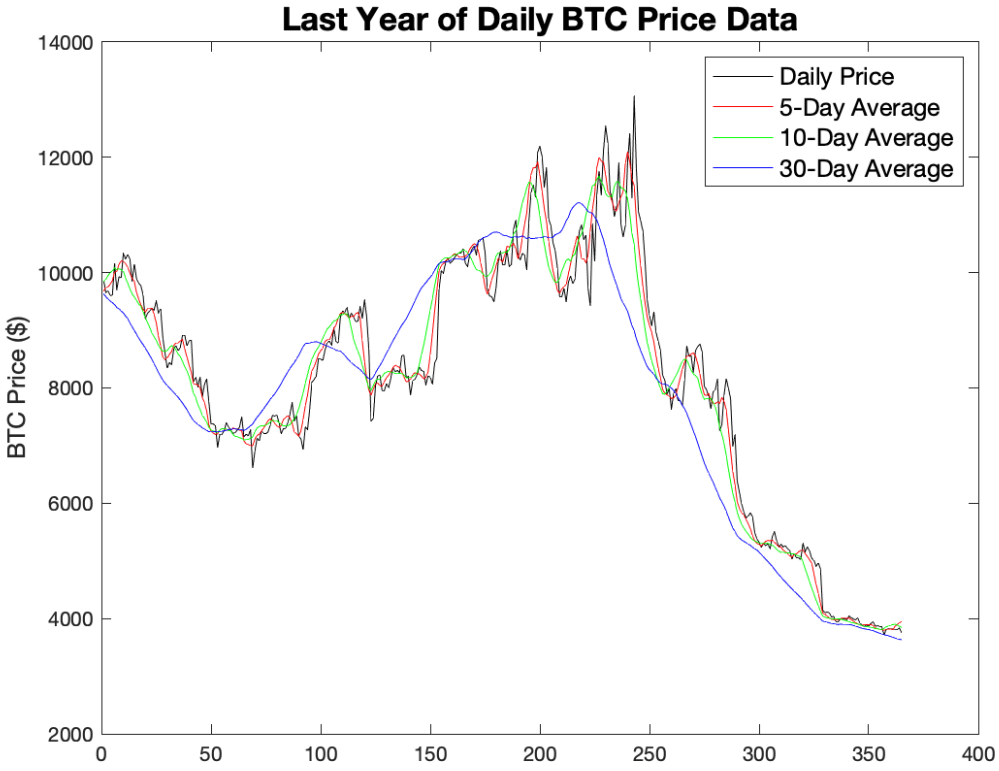
1) Start by defining the main function with no inputs and no outputs. Load the BTC data from inside the function rather than passing it in.
   - This way you won't have to go back and forth between a test script while you are testing/debugging (i.e. you can just click the run button).
   - Add a debug point at the end of the main function so you can inspect the local workspace after the code has been executed.
2) Reduce the data to the time period interest.
   - To plot the 30-day moving average for the last 365 days, we will need the last 394 days of price data.
3) Create a loop to calculate the moving averages. Initialize the storage vectors first using the `zeros` function.
4) Plot the results. Annotate appropriately.
5) Redefine the main function to accept two inputs and no outputs. Move the code that loads and reduces the BTC data into a separate script.
6) Test your function from the new script. Try passing in different time periods to your function and ensure that it still works as expected.

Function Specifications:

Function Name	Purpose	Input(s)	Output(s)
`movingAverages`	plot price and moving averages for a given stock or cryptocurrency	1. string array containing dates 2. vector containing prices	none

Notes:

- The first element in the `prices` array is the most recent date. It makes more sense to flip the arrays when plotting so the x-axis values are in order from oldest to most recent.
- The x-tick labels will be numbers, which isn't very useful. If you correctly insert dates, you'll recieve extra credit.
- The default axis limits will probably not look great. Relate the axis limits to the time period of interest to add some white space at the beginning and end of the plot for extra credit.



Full Credit (25 points)

Full+Extra Credit (35 points)

Useful functions: `plot`, `axis`, `xlabel`, `ylabel`, `legend`, `flip`

Deliverable(s): `movingAverages.m`

Graded Item	Point Value
Adequate comments and organization	5
Correct input/output	20
Total	25

Problem 5

Open a new MATLAB file (or start from our function template) and create a new main function called `calcAllGrades` that accepts a single input and a single output, both of which will be structures. Copy the `calcGrade3` function from Lab 4 and paste it into the new file below the main function. Rename the new subfunction to `calcGrade`.

For this problem, you will loop through the entire structure and calculate the grade for each student. Just before calculating a student's grade in the loop, you will check to make sure the student's structure has the all the required fields and that they are all numeric.

In addition to the average grade and letter grades that we determined in labs 3 and 4, you will drop the two lowest quizzes and the lowest lab grade, and assign them to the fields `dropAvgGrade` and `dropLetterGrade`, respectively.

There are several parts to this problem. Don't try to do too much at once! You are free to approach this however you like, but my suggested steps are listed below.

1) Start by defining the main function with no inputs and no outputs. Load the grades structure from inside the function rather than passing it in.
   - This way you won't have to go back and forth between a test script while you are testing/debugging (i.e. you can just click the run button).
   - Consider adding a debug point at the end of the main function so you can inspect the local workspace after the code has been executed.
2) Create a loop in the main function that will print out each field in the main structure using `fprintf`.
   - Once you have this simple loop written, you can access the fields with those strings.
3) Remove the `fprintf` statement and use the `calcGrade` function to calculate the grades of each student.
   - No modifications need to be made to `calcGrade` yet. You should use the function exactly like you did for Lab 4.
   - At this point, you should have all 500 students' averages and letter grades stored in the main structure after the loop has completed.
4) Move the letter grade conditional statements to a separate function (I called mine `determineLetter`).
   - The function should take the average grade (a 1x1 double) as input and output the letter grade (a 1x1 string).
5) Copy and paste the `calcGrade` function at the end of the file. Modify the new function to drop the lowest two quizzes and the lowest lab grade before calculating the average.
   - This function should work almost exactly like `calcGrade` and should be called from the main function immediately before or after `calcGrade`.
6) Create a function that ensures each student structure has the required fields and that the value for each field is numerical.
   - This function should be called from the main function immediately before calculating grades.
   - The function should take a student structure as input and output a logical true or false (1 or 0).
   - Use the logical output from this function to determine whether or not to proceed with the grade calculations.
   - If the output is false, display a warning message and continue to the next student. If the value is true, calculate the current student's grade.
7) Redefine the main function to accept a single input and a single output.
8) Add a check at the beginning of the main function to ensure that a single input is provided. If zero or more than one is provided, display an error message and terminate execution.
9) Pat yourself on the back, you just finished writing a relatively robust grading program.

Function Specifications:

Function Name	Purpose	Input(s)	Output(s)
`calcAllGrades`	calculate grades for all students in a structure	structure with any number of student substructures, each containing 4 fields (`exams`, `labs`, `quizzes`, `projects`)	structure with any number of student substructures, each containing 8 fields (`exams`, `labs`, `quizzes`, `projects`, `avgGrade`, `letterGrade`, `dropAvgGrade`, `dropLetterGrade`)

Useful functions: `isfield`, `isnumeric`, `warning`, `fprintf`

Deliverable(s): `calcAllGrades.m`

Graded Item	Point Value
Adequate comments and organization	5
Correct input/output	20
Total	25