Melbourne School of Engineering

Engineering Systems Design 2

Assignment Five

Learning Objectives

In this assignment you will demonstrate the ability to:

- Write functions to perform the appropriate tasks.
- Demonstrate the appropriate use of programming techniques such as computation, selection and repetition.
- Demonstrate the ability to write programs to perform low-level input/output.
- Test your code to ensure that they produce the correct output.

Assessment

- This is an individual assignment that will contribute towards 3% of your assessment in this subject.
- For each question, you must submit only one MATLAB function file (.m file), which has the exact name and signature (using the correct case of letters in the name of the function) as specified in the question. If you happen to write more than one function for the question (this is not really required for the assignment questions we have provided), include them in the same file that contains the main function (the function that is specified in the question).
- For each function, you will be provided a test function, which you must use to do some 'basic sanity testing' before you submit. The name of this function will always be:

  `[functionName]StudentTest.m`

  For example, if the function you are required to test is `computeDistance` the test file that is provided is:

```
computeDistanceStudentTest.m
```

Place this file in the same directory where your file for the function is, and execute the following command:

```
run([functionName]StudentTest)
```

In the above example,

```
run(computeDistanceStudentTest)
```

and this will run the basic tests cases we provide and report the status.

The following shows an example output from the test run.

```
>> run(computeDistanceStudentTest)
   Running computeDistanceStudentTest
   .....
   Done computeDistanceStudentTest
   ----------


   ans =

     1x2 TestResult array with properties:

       Name
       Passed
       Failed
       Incomplete
       Duration

   Totals:
      2 Passed, 0 Failed, 0 Incomplete.
      0.050619 seconds testing time.

   >>
```

Note that this script will only do some basic testing, and your function must pass at least these test cases before submission. However, during assessment we will be running more test cases and, therefore, your testing should not be limited to this test run.

- Ensure that the code is well commented your comments should be detailed enough to show your understanding of how your functions work and this forms part of how you will be marked for this assignment. You will also be marked on the readability and efficiency of your code (i.e. using loops where appropriate).
- You must strictly follow the instructions for each question. For example, if the question specifically states that you are required to use loops rater than built-in MATLAB functions, and you do not follow this, you will receive 0 marks for the output even if the output is correct.

Submission

Download the file, `AssignmentFive.zip`, which has the student test cases and data files for all the questions from LMS. Unzip this file and a directory `AssignmentFive` will get created (this directory will have the .m files with test cases as per naming convention described above). Place the .m files for your solutions in `AssignmentFive` directory. Once you complete your assignment, create a .zip file of the `AssignmentFive` directory, which will have the name `AssignmentFive.zip`. Submit this .zip file via LMS by the submission deadline.

Make sure you follow the instructions in this specification carefully, because we will be running automated testing during assessment, and the tests will fail if you do not strictly follow the instructions.

Questions

Question 1 [15 Marks]

Write a function `nybblise` which takes as input digital signal of 0's and 1's stored in a vector of size $N \times 1$, and breaks it into four 4-bit nybbles, which are returned as the columns of a $4 \times N/4$ matrix. For example, the 16 bit input should be mapped as:

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{16} \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} d_1 & d_5 & d_9 & d_{13} \\ d_2 & d_6 & d_{10} & d_{14} \\ d_3 & d_7 & d_{11} & d_{15} \\ d_4 & d_8 & d_{12} & d_{16} \end{bmatrix}$$

Your function must return -1 if the N is not a multiple of 4.

Following is the signature of the function.

```
function A = nybblise(d)

Input:
    d - digital signal as a vector of size N x 1

Output:
    A - nybbles as a 4 x N/4 matrix OR
                        -1 if N is not a multiple of 4
```

Note: This function must use loops rather than MATLAB built-in functions such as `reshape`, `fliplr`, `flipud`. If you do not use loops, you will not receive any marks for the test output, even if test cases pass.

Question 2 **[10 Marks]**

Write a function `computeMatrixMax` that will take a matrix as input, and return the maximum values of each row (as a vector) and the maximum value of the entire matrix. Following is the function signature.

```
function [row_max matrix_max] =  computeMatrixMax(A)

Input:
     A - a is a matrix (the size is arbitrary)

Outputs:
    row_max - a vector that contains the maximum value of each row
    matrix_max  - the maximum value of the matrix
```

For example if,

```
A =

     1     6     5
     2     4     7
```

then the function should return:

```
row_max =
     6     7
```

```
matrix_max =
     7
```

Note: You must use loops to iterate through the matrix and compute the required values, rather than using MATLAB built-in functions such as, `max`. If you do not use loops, you will not receive any marks for the test output, even if test cases pass.

Question 3 **[15 Marks]**

The high temperature for each month for a year (in Fahrenheit, rounded to integers) for different locations are stored in a file. Each line of the file has a location ID, followed by 12 temperatures. For example, the file might store:

```
432 33 37 42 45 53 72 82 79 66 55 46 41
777 29 33 41 46 52 66 77 88 68 55 48 39
567 55 62 68 72 75 79 83 89 85 80 77 65
```

The value on the first field in each row (432, 777 and 567 in this example) is a location ID. The 12 numbers that follow the location ID are the high temperatures for the months.

Write a function `computeMaxTemperature`, that will take the filename as an input argument, compute the maximum temperature for each location, and return two row vectors: the first vector must contain the location IDs in the same order they appear in the file; and the second vector must contain the top temperature for each location (the same order as the locations). You must use the `computeMatrixMax` function from the previous example.

If the specified file does not exist, the function must return -1 for both the location ID and the maximum temperature. If the specified file exists, you can assume that the data in the file has the correct format.

Your function will have the following signature:

```
function [loc_id max_temp] = computeMaxTemperature(file_name)

Inputs:
   file_name - name of the file that contains the data


Output:
  loc_id  -  1 x N vector with location IDs, in the same order they
             appear in the file - N is the number of rows in the file
```

```
            -1 if the file does not exist
  max_temp - 1 x N vector that contains the maximum temperature for
            each location
            -1 if the file does not exist
```

In the above example, if the file specified above was given as the input file name, the output must be:

```
loc_id = [432 777 567]
max_temp = [82 88 89]
```

Question 4 **[5 Marks]**

Biomedical engineers are developing an insulin pump for diabetics. To do this, it is important to understand how insulin is cleared from the body after a meal. The concentration of insulin at any time is described by the equation:

$$C = C_0 e^{\frac{-30t}{m}},$$

where C_0 is the initial concentration of insulin, t is the time in minutes, and m is the mass of the subject in kg.

Write a function, `computeInsulinConc`, which takes as input, the initial concentration C_0, the mass of the subject m in kg, and a vector of time values, t which gives the elapsed time since the of the initial concentration of insulin, and returns the insulation concentration as the time values specified in the vector t.

The signature of the function is:

```
function c = computeInsulinConc(C0, m, t)

Inputs:
     C0 - initial insulin concentration
     m  - mass of the subject in kg
     t  - vector of time values of elapsed time since the of the initial
          concentration of insulin in minutes

Output:
     c  - a vector of insulin concentration values at the times
          specified in t
```

Question 5 **[15 Marks]**

Write a function, `plotInsulinConc` that will plot on the same graph, the insulin concentration (as per specifications in the previous question) vs elapsed time for different weights, over a specified period of time; your function must save the plot in JPEG format. The graph must have the appropriate legends to show which line corresponds to which subject weight.

The function will have the following signature:

```
function plotInsulinConc(C0, M, t_start, t_end, t_step, file_name)

Inputs:
   C0 - initial insulin concentration
   M - this is a 1 x N vector that contains the mass of the
        subjects in kg
   t_start -  start time (minutes)
   t_end - end time (minutes)
   t_step - the plot should compute points at time step
            defined by this value (minutes)
   file_name - name of the file to save the plot
               (without the file extension)
```

For example, if `t_start` = 0, `t_end` = 10 and `t_step` = 1, the values concentration must be computed and plotted at time values: `0 1 2 3 4 5 6 7 8 9 10`. If the `file_name` is `plot_1` the graph must be saved in a file named `plot_1.jpg` Figure 1 shows an example of the required plot.

The function must:

- Use `computeInsulinConc` function from the previous question.
- Label the axis and include the legend as shown in the example output in Figure 1. Hint: store the legend labels in a cell array while iterating through the loop.

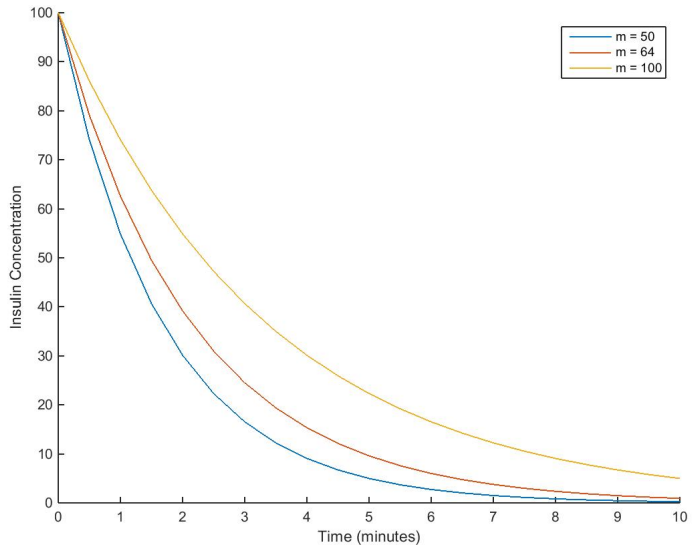

Figure 1: Insulin Concentration Plots for different weights.