Practical Problems 3 – Data Structures, Formatting & Functions

Create a single script file for questions 1 - 4.

1. Create a cell array called ‘myCell’ where the first element is a random 3 x 4 matrix of integers, the second element is a vector with 200 elements ranging from -10 to 10, and the third element is a string of your choice.

2. Request the user input for an entry in the 3 x 4 matrix. Your code should check that the user entered a valid pair of numbers and display a warning message if not.

3. Create a new vector that is the vector in the cell array multiplied by the number in the matrix specified by the user, then insert this new vector as the third element of the cell array and shift the previous third element (string) to the fourth position of the cell array.

4. Plot the 2 vectors from your cell array against each other in one line of code.

Create a single script file for questions 5 - 10.

5. Create a structure called myStruct to save the following data:

Name	Age	Height (cm)	Mass (kg)
Harry	36	170	80
Georgia	21	181	70
Elizabeth	78	158	65

6. Change Georgia’s mass to be 68 kg.

7. Add an extra person to the data with the following details: Name, Lily; Age, 24; Height, 162 cm; Mass, 60 kg.

8. Using one line of code, calculate the mean height of the group.

9. BMI is given by Mass (kg) divided by Height (metres) squared. Calculate the BMI of each person in the group and make a new field in your structure that saves this data for each person.

10. Use the sprintf function to display the names of the people in the group along with their age, height, mass and BMI displaying each value to 2 decimal places).

11. Write a script that displays the number 12345.987654321 in the following formats (include leading 0's if the width is longer than the number):

    (a) Width 10 precision 3.

    (b) Width 10 precision 6.

    (c) Width 8 precision 6.

    (d) Width 14 precision 6.

    (e) What happens if you use a higher precision than there are decimal places (e.g. precision 12)?

12. Write a script that finds and displays the largest x that can be input to the exponential function (e^x) before an infinite result is reached.

13. Write a script that asks the user for 2 inputs. Prompt the user for a random sentence. Then prompt the user for a letter of the alphabet. Your script should then display how many times that letter appears in their sentence and lists the words in which it appears.

14. Load the datafile "DOB.mat" into the Matlab workspace then create a categorical array of the generations defined as follows:

    "Boomer" born in 1946-1964
    "Gen X" born in 1965-1980
    "Millennials" born in 1981-1996
    "Gen Z" born in 1997-1997-2012
    "Gen A" born in 2012-present

    (Hint: Use the discretize() function)

    Display how many of each generation are in the data set then create a table that contains columns for each day, month, date, year and generation.
    Display the first 10 table entries.
    Now display the date of births of all Gen A.
    Lastly, locate all people born in July and display the corresponding year of their births, the day on which they were born, and their generation.

15. The following code performs some tasks in Matlab using a for loop. Vectorise the code to produce the same result without using any loops.

```
tstart=0; tend=20; ni=8;
t(1)=tstart;
y(1)=12 + 6*cos(2*pi*t(1)/(tend-tstart));
for i=2:ni+1
  t(i)=t(i-1)+(tend-tstart)/ni;
  y(i)=12 + 6*cos(2*pi*t(i)/ ...
       (tend-tstart));
end
```

16. Create a random 4 x 4 matrix, ***A***, with values between 0 and 1 then swap the 2nd and 4th row using one line of code.

17. Using the matrix, ***A***, from Q13 now swap the 1st and 4th columns using one line of code.

18. Using one line of code, set every value of ***A*** that is greater than 0.5 equal to 7.

19. Create a 4 x 7 matrix, ***B***, of random integers between 10 and 50 then using one line of code set every value of ***B*** that is both greater than 20 and less than 40 equal to 0 (***Hint:*** Read about '&' and '|' logical operators in the documentation).

20. Using one line of code insert the column vector $[3\ -2\ 4\ 8]^T$ in between the 2nd and 3rd columns of ***B***.

21. Write a script that creates an anonymous function for the function $f(x) = x^2 - 3x + 1$ then plots it between -4 and 4 with a grid.

22. Write a script that creates 2 anonymous functions, f1 and f2, for $\sin(x)$ and $\cos(x)$, then a 3rd anonymous function, f3, equal to the first function divided by the second function (f1/f2 = $\sin(x)/\cos(x) = \tan(x)$). The script should then plot all 3 functions in different colours using subplots over the domain $-2\pi < x < 2\pi$. Set the axes on the graph to have y-limits between -1 and 1.

23. Create an anonymous function, f4, that takes 2 inputs, x and y, then calculates x^y. Test your function with inputs x = 3 and y = 7.

24. Create an anonymous function, f5, that takes 3 inputs, f, a and b, where f is another anonymous function, and plots f between a and b. Test f5 by using any test function over whatever domain you like.

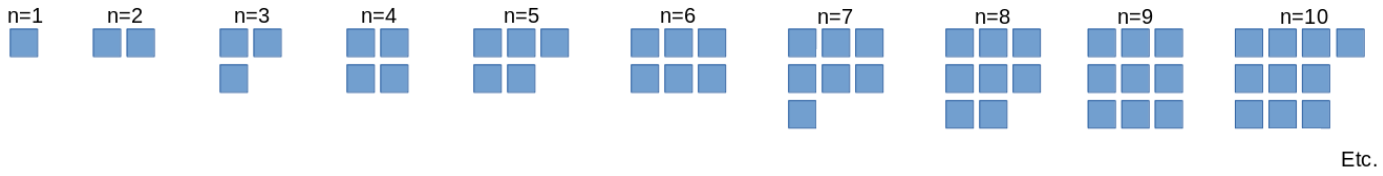
25. Create a function file (**not anonymous**) that takes 3 inputs and produces a surface plot. The first input should be a function that accepts 2 arguments (x and y), the second and third inputs should be vectors containing the desired values of x and y over which to plot. Test your function using any function f4 and domain.

26. Write a function file that takes between 3 and 6 input arguments (all scalar values) and returns a vector containing all the inputs sorted from lowest to highest. Warning messages should be displayed in the command window if the user enters the wrong number of inputs, or a vector/string instead of a scalar.

27. ## Challenge Problem

Write a function file that requires 3 inputs, *F, a* and *b*, where *F* is a function of one variable, and plots the function between *a* and *b*, but also accepts additional optional input arguments that specify other intervals to plot the function over.

The plots should appear in the same figure (subplots) with the pattern as shown below:



Note that the additional optional arguments should come in pairs (*c* and *d*, *e* and *f etc.*). Display warning messages if the user enters the wrong number or type of input. Finally accept up to 100 subplots.
Hint: To determine if the first argument is a function read about **isa()** in the documentation.
Hint: To determine the layout of the subplots it will be helpful to think about the relationship between square numbers, number of plots requested (n), and the number of rows and columns.