Introduction to Programming 2017-2018

Exercise 7

Question 1

A parabola function can be described by the formula: $\boldsymbol{y = ax^2 + b}$.

Write a function called '**parabFunc2'** that calculates the values of *y* for a given set of parameters (*x*, *a* and *b*), according to the following:

1. The function should get one mandatory input argument – a vector of numbers which will be used as the *x* values in the parabola formula.
2. The function may get up to two more optional input arguments (according to this order of appearance):
   a. A scalar which will be used as the '*a*' parameter in the parabola formula. If this value was not provided the function should calculate the parabola formula with the default value of: *a*=1.
   b. A scalar which will be used as the '*b*' parameter in the parabola formula. If this value was not provided the function should calculate the parabola formula with the default value of: *b*=0.
3. The function should return one mandatory output argument – the vector of *y* values (parabola function values) calculated for the corresponding values of x in the input vector, using the appropriate *a* and *b* parameters.

Note: In this function do not use varargin (just nargin).

Examples:

xxVec=[-3:1:3]

outVar = *parabFunc*(xxVec) ➔outVar=[9, 4, 1, 0, 1, 4, 9]

outVar = *parabFunc*(xxVec,-2) ➔outVar=[-18, -8, -2, 0, -2, -8, -18]

outVar = *parabFunc*(xxVec,0.5, 3) ➔ outVar=[7.5, 5, 3.5, 3, 3.5, 5, 7.5]

Question 2

Write a function called '**parabFunc3'**, which does the same as parabFunc2, but instead of using nargin, use just varargin.

Question 3

A magic square of order *n* is an arrangement of n^2 numbers, in a square, such that the numbers in all rows, columns, and both main diagonals sum to the same constant (see examples).

The constant sum of every row, column and diagonal is called the magic constant, $M = \frac{n(n^2+1)}{2}$.

In the cold winter, the squirrels love to calculate whether squares are magical.

Write a function called **magicCheck** that will help the mathematically challenged squirrels.

The function receives **one** input argument:

1. **someSquare** (*matrix n*n*) – square matrix with integer numbers.

The function returns one output argument, **ansCell** (*cell array 1*3*) with the following elements.

1. The string ‘yes’ or 'no', depending on the square’s magic properties.
   - The square’s magic properties should be <u>calculated within an internal function</u>, called: **mCalc.** This function should get one input argument – **matData** – matrix (the same matrix as in someSquare),

     This function returns one output argument – **mAns -** The string ‘magic square’ or 'no magic', depending on the square’s magic properties.
2. A vector with the sum of the matrix **columns**.
3. A magic square contains the integers from 1 to n^2. If a larger number is found this string should be 'Too big', otherwise it should be 'OK'.
   - This value (for part 3) – should be <u>calculated within an internal function</u> called **bigCalc.**

     This function should get one input argument – **matData** – matrix (the same matrix as in someSquare),

     This function returns one output argument – **bigAns -** The string 'Too big' if a larger number is found, or 'OK' otherwise.

The main function (magicCheck) creates a new figure that shows a bar graph with the mean of all the **columns**. Add: a title, x-label and y-label to the graph.

Example 1:

$$\text{myMat} = \begin{bmatrix} 1 & 4 & 14 & 15 \\ 4 & 7 & 8 & 6 \\ 2 & 1 & 8 & 2 \\ 3 & 15 & 6 & 13 \end{bmatrix};$$

[myCell] = magicCheck(myMat)

myCell{1,1} = 'no' myCell{1,2} = [10 27 36 36] myCell{1,3} = 'OK'

Example 2:

$$\text{myMat} = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix};$$

[myCell] = magicCheck(myMat,rowN)

myCell{1,1} = 'yes' myCell{1,2} = [34 34 34 34] myCell{1,3} = 'OK'

General notes:

- Submission date: Soft copy should be submitted by **Monday (8.1.18) / Wednesday (10.1.18) 23:59**. Hard copy should be submitted at the beginning of your Tirgul group.
- Make sure to write the student name and id number as a comment in the code.
- Document your functions using comments in the code. You can add comment throughout the code as you see necessary, but the following comments are required:
- At the beginning of each function, explain what the function does and describe the input and output arguments.
- Add comments in the code before loops and before logical conditions, and when creating new variables.

Good luck!