MATLAB Programming NCTU Fall 2017 Exam#2 2018/1/8

You need to write a m-file for each problem. Name your m-files **P1.m**, **P2.m**, etc.

You can consult the lecture slides, your own class notes, MATLAB documentation, and other printed material you can bring with you, but no other material is allowed. You are not allowed to talk with one another or use your own 3C devices or USB drives. You can use any function that we have covered in the class unless noted otherwise, or any standard MATLAB functions you can find. However, you cannot use toolbox functions that are not covered in the class.

1. [35%] The task of this problem is to implement a GUI program that draws a list of (x,y) points. Name a program **`P1`**. The points are stored in a MAT file (named **`cF2017finalP1.mat`**) that contains a single variable **`pt`**. Each row of **`pt`** represents a point. More details are described below:

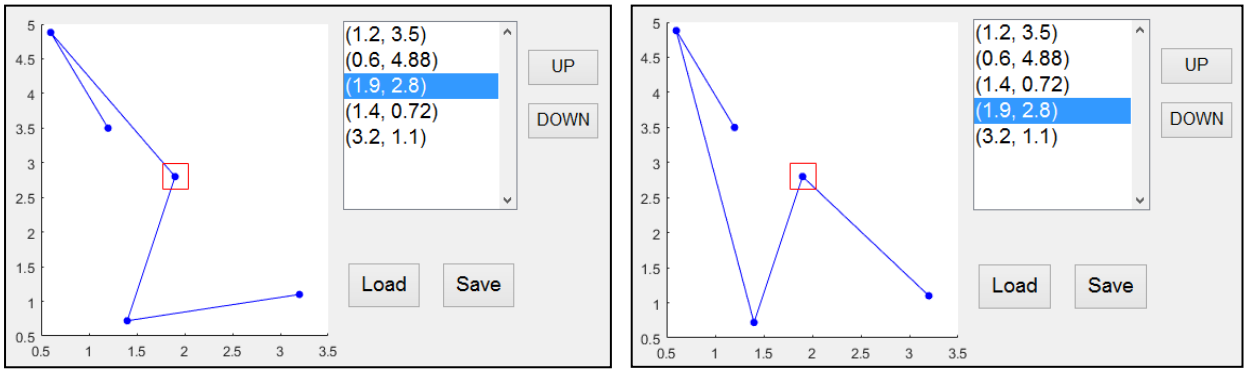
(1) Click **`Load`** to read the file (to be supplied at the time of the exam). Fill the list box with the coordinates of the points. Use the format as in the example.

(2) In the beginning, the first point is selected.

(3) Draw line segments that connect the points in the axes. The selected point is highlighted with a rectangle. (If the user selects a different point, redraw the plot to highlight the newly selected point.)

(4) The user can change the ordering of the points using the **`UP`** and **`DOWN`** buttons. The selected point is moved up or down by one position if applicable. The selected point should remain selected after the movement, meaning that you need to modify the **`Value`** property of the list box. Redraw the plot to show the new ordering. (The example illustrates the effect of a user selecting the third point and then clicking **`DOWN`** to move it down.)

(5) If the user clicks **`Save`**, write the points, with the current ordering, into the MAT file. The next time you click **`Load`**, the points should be in the last saved order.



2. [35%] Design a MATLAB class **`P2`** to represent arrays of circles.

There are three properties:

- **`x`**, **`y`**: the center coordinates
- **`r`**: the radius (always >= 0)

Constructor:

- **`nargin`**==0: A default scalar circle (**`x`**=0, **`y`**=0, **`r`**=1)
- **`nargin`**==3: The three inputs represent **`x`**, **`y`**, **`r`**; these inputs have to be arrays of the same size. The output is an array of circles. (Stop on error if any element of **`r`** is negative.)

Overloaded methods/operators:

- **`le`** (operator **`<`**): Pairwise comparison of radius. The two inputs have to be arrays of the same size, or one array and one scalar. The output is a logical array. Use no loops for this function.
- **`set.r`**: This is for you to ensure that any direct assignment to **`r`** is non-negative. For simplicity, this function should only work with a scalar object. Output an error message if the object is not scalar, or if

the radius given is negative.

- **disp**: Each circle is listed in the form of "(x,y,r)". Display an array in matrix form. (We will test only 2-D arrays.)

Additional method:

- **iswithin**: The first input is an array of circles, and the second input is a 2-element vector representing the (x,y) coordinates of a point. The output is a logical array (the same size as the first input) indicating whether the point is with the circles. Use no loops for this function.

Example:

```
C = P2
(0,0,1)
C = P2([0 1; 2 3], [1 0; 4 3], [2 3 1 5])
(0,1,2) (1,0,3)
(2,4,1) (3,3,5)
(C < P2(6,1,4))
     1   1
     1   0
C(1,2).r = 2
(0,1,2) (1,0,2)
(2,4,1) (3,3,5)
C(1,2).r = -1
error
iswithin(C, [0 2])
     1   0
     0   1
```

3. [30%] Design a function that does multi-variable polynomial evaluation. Name the function **P3**. The polynomial is represented as an array where each element is a coefficient. Each dimension of the array corresponds to a variable.

Example: The coefficient array for $3x^2y^2 - 2x^2y + 4xy^3 + 2xy + 5y^2 - 1$

is

```
0   3  -2   0
4   0   2   0
0   5   0  -1
```

For simplicity, this function only needs to do evaluation at a single point. There are two input arguments: The first argument is the coefficient array, and the second argument is a vector containing the values of x, y, etc.

Example result: `P3([0 3 -2 0; 4 0 2 0; 0 5 0 -1], [2 -1])` is 12.

Example result (three variables: $xy^2z^3 + 2y - 3z$):

```
p(1,1,1) = 1;  p(2,2,4) = 2;  p(2,3,3) = -3;
P3(p, [3 2 -2]) is -86.
```

You can use a loop over the dimensions.

If you can only get a function that works correctly for two-variable polynomials (no loops), you can get a maximum of 24 points.