MATLAB Programming NCTU Spring 2017 Exam#2 2017/6/16

1. [35%] Design a MATLAB class **`P1`** to represent arrays of rectangles.

There are five properties:

- **`x1`**, **`y1`**, **`x2`**, **`y2`**: the boundary coordinates
- **`empty`**: (logical) whether this object is an empty rectangle

Constructor:

- **`nargin`**==0: a default scalar empty rectangle (flag **`empty`** is true)
- **`nargin`**==4: The four inputs represent **`x1`**, **`y1`**, **`x2`**, **`y2`**; these inputs have to be arrays of the same size. The output is an array of rectangles.

Overloaded methods/operators:

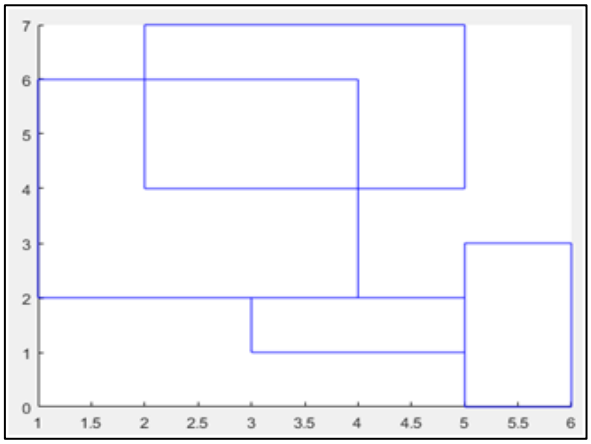
- **`and`** (operator **`&`**): Pairwise intersection of rectangles; the two inputs have to be arrays of the same size. For example, the intersection of (0,2,4,4) and (1,3,2,5) is (1,3,2,4). If an intersection is empty, the corresponding output should be an empty rectangle (flag **`empty`** is true).
- **`or`** (operator **`|`**): Pairwise union of rectangles; the two inputs have to be arrays of the same size. For example, the union of (0,2,4,4) and (1,3,2,5) is (0,2,4,5). The union of a rectangle with an empty rectangle is always the original rectangle.
- **`disp`**: Each rectangle is listed in the form of "(x1,y1,x2,y2)". Display an array in matrix form. (We will test only 2-D arrays.) An empty rectangle is displayed as "<empty>". Don't worry about alignment.

Additional method:

- **`draw`**: Create a figure and plot all the rectangles (see the example). Skip empty rectangles.

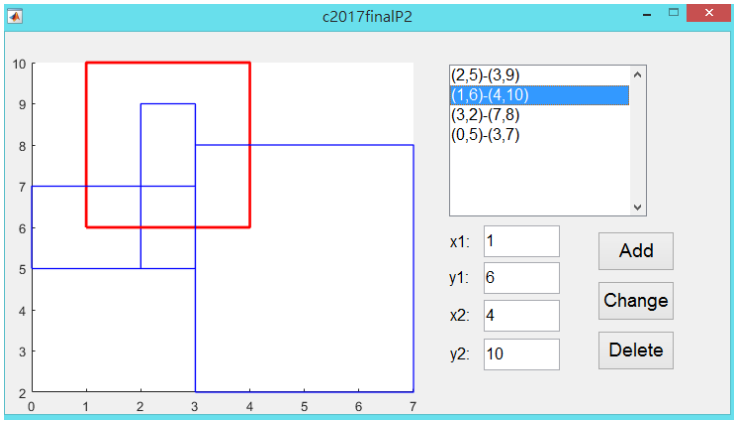
The following is some test sequence:

```
>> r = c2017finalP1
r =
<empty>
>> r = c2017finalP1([1 5; 2 3], [2 0; 4 1], [4 6; 5 4], [6 3; 7 2])
r =
(1,2,4,6)   (5,0,6,3)
(2,4,5,7)   (3,1,4,2)
>> a = r(1,:) & r(2,:)
a =
(2,4,4,6)   <empty>
>> a = r(1,:) | r(2,:)
a =
(1,2,5,7)   (3,0,6,3)
>> r.draw
>> a(2,2) = c2017finalP1(1, 3, 5, 7)
a =
(1,2,5,7)   (3,0,6,3)
<empty>   (1,3,5,7)
```



2. [35%] Create a GUI program **`P2.m`** for displaying a set of rectangles. A sample is shown here. Each rectangle is specified by four coordinates **`(x1,y1,x2,y2)`**.

- The user can add rectangles using the edit boxes and the **`Add`** pushbutton.
- The list box lists all the rectangles in the format of "(x1,y1)-(x2,y2)".
- When a rectangle in the list box is selected, list its corner coordinates in the edit boxes.
- When a rectangle in the list box is selected, highlight that rectangle in the plot using either a different color, a thicker line width, or both.
- When the user just adds a new rectangle, that rectangle is selected and highlighted.
- The user can change the selected rectangle using the edit boxes and the **`Change`** pushbutton.
- The user can delete the selected rectangle using the **`Delete`** pushbutton.
- You need to maintain an internal data item to remember all the rectangles.



3. [30%] Design a viewer/selector program of images using the simple form of user interaction: Use only keyboard and mouse clicks, and no GUI objects.

- In the beginning of the program, use **`dir('*.jpg')`** to read the list of image file names. Use **`imshow`** to display the first image.
- Use keys '**`n`**' and '**`p`**' for moving to the next and previous images, respectively. Show the image file name and its index in the list in a text right above the image, as in the example.
- Draw a small square near the top-left corner when showing the image to indicate whether the image is "selected". You can use a filled square for a selected image, and an unfilled square for an unselected image. (All the images are unselected in the beginning.)
- When the user clicks inside the square, this toggles the state of selection. You need to remember the state for each image. This means that when you visit the image next time, its previous state of selection is preserved.