Write a function that takes an integer **N** as input and draws a diamond shape of size **N**. Hint: Use a 2-D array of characters. Examples are shown below:

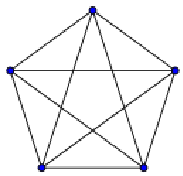
```
N=1     N=2     N=3     N=4     N=5     N=6     N=7
*       **       *       **       *       **       *
        **      * *     *  *     * *     *  *     * *
                 *      *  *    *   *   *    *   *   *
                         **      * *    *    *  *     *
                                  *      *  *    *   *
                                          **      * *
                                                   *
```

Write a function that draws a filled rectangle in a RGB image. The function can take two or three inputs. The header of the function looks like

```
im = f(im, rect, clr)
```

Here **im** is a RGB image, **rect** is a four-element vector **[x1, x2, y1, y2]** indicating the boundary coordinates of the rectangle, and **clr** is a three-element vector representing the color of the rectangle.
The input **clr** is optional. If it is omitted, the default color is white.
You can assume that **im** is of type double.
At the end of your function, use **image** or **imshow** to display the image.
Your function should be able to handle cases when **x2<x1** or **y2<y1**, and when any of these coordinates are outside of the image.

Write a function that takes an integer **n** as its only input. This function draws a complete graph of **n** vertices, with all the vertices equally spaced along a circle with r=1 and centered at the origin. For example, if **n**=5, the plot should look like the one shown to the right. (If you use any loops, you'll get no more than **15** points for this problem.)



Write a function that takes a 2-D array and an integer **n** as its inputs. The output is another 2-D array, where each element in the input array is expanded to a **nxn** block. An example is shown below. (If you use any loops, you'll get no more than **15** points for this problem.)

```
disp( f([0 1 2; 3 4 5], 3) );
  0     0     0     1     1     1     2     2     2
  0     0     0     1     1     1     2     2     2
  0     0     0     1     1     1     2     2     2
  3     3     3     4     4     4     5     5     5
  3     3     3     4     4     4     5     5     5
  3     3     3     4     4     4     5     5     5
```