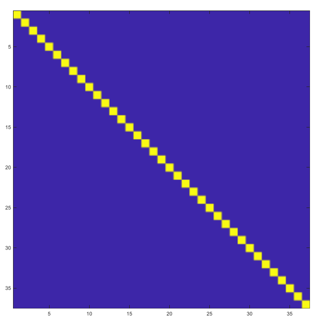
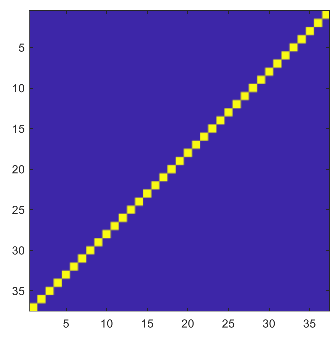
COM322 LAB 2: MATLAB INTRO CONTINUED

Write code for the following problems in the MATLAB editor and save the source files as m files (with .m extensions).

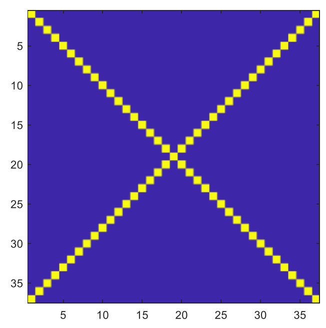
1. Write code that produces the following figures. Hint: fill an array with zeros and ones and display the array using imagesc.



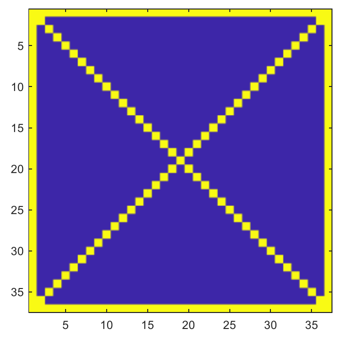
a)



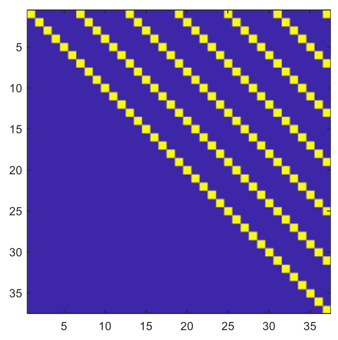
b)



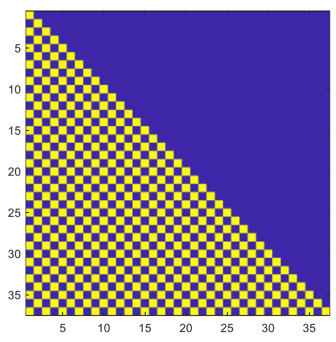
c)



d)

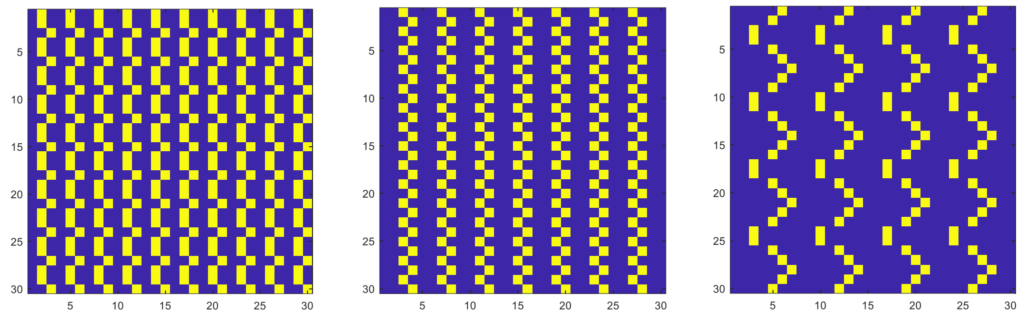


e)



f)

2. Write a function called **divN** that takes 3 numbers, **r**, **c**, and **N** as parameters and returns 1 if $\mathbf{r^4}+\mathbf{c}$ is divisible by **N**, otherwise returns 0. Write another function called **main** which will call **divN** for every row (**r**) and column (**c**) pair of a square matrix of size 30x30, assign the result of the call to that entry and display the result in a figure using imagesc. Assign a fixed value for **N** during each run. The resulting patterns for various values of **N** are shown below (3, 4 and 7 from left to right). *Hint*: use the modulus function called **mod** to check if a number is divisible by N.

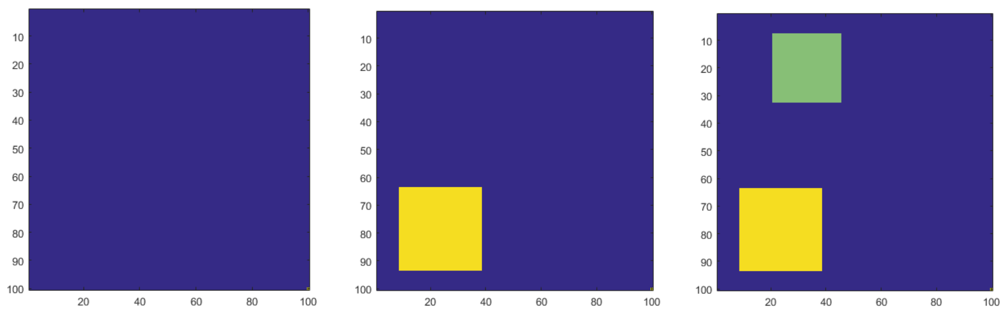


3. Write a function that will assign values into a square submatrix of a matrix passed as parameter. The function will return the new matrix. Use the following function declaration:

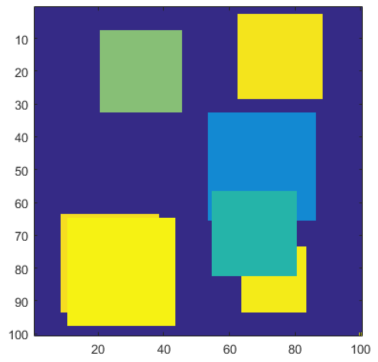
```
        function A = square(A, r, c, size, intensity )
```

where the **A** argument is the original square matrix, **A** as the return value (on the left) is the modified matrix being returned, **r** is the lowest row index of where the values given by the parameter **intensity** will be assigned, **c** is the lowest column index and **size** is the number of indices on one side of the submatrix being assigned the given values.

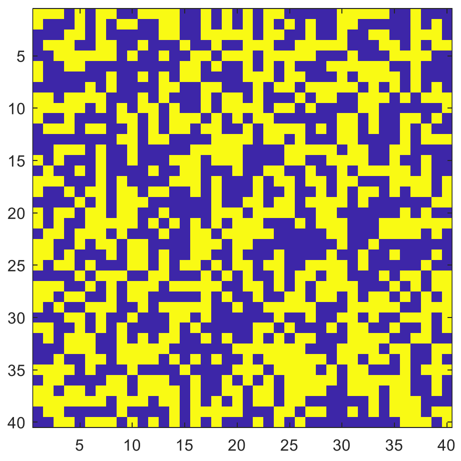
The figures below show how the function affects the contents of a matrix. Here we start with A (shown on the left) filled with zeros. When the function is called for the first time, matrix A is then passed into the function and the modified matrix is returned (middle). If we repeat the process, a second invocation of the function will insert another square into the matrix (right) thus accumulating the changes.

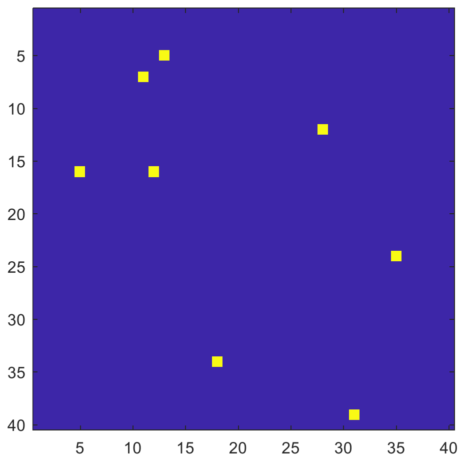


Now write a main function that will initialize a 100x100 matrix (**A**) with zeros and then call **square** 7 times to accumulate the results of the function in the same matrix. For each call, starting points, sizes and intensities should be randomized. Make sure that the squares fit into the 100x100 area and don't run off the sides. Display the result using imagesc. An example run is shown below.



4. In this part you will write a program to perform a simple analysis of a random image. First, generate a random array of size 40x40 with each entry being either 0 or 1 and display it in Figure 1. Next, write a program segment that searches for single entries (value = 1) that have no neighbors (all 8 neighbors have values of 0) and records them in another array of the same size. Display the results visually in Figure 2. An example run is shown below: content of random matrix (left) and entries with no neighbors (right).


5
10
15
20
25
30
35
40
5
10
15
20
25
30
35
40


5
10
15
20
25
30
35
40
5
10
15
20
25
30
35
40