2) Write a matlab function which has the following properties:

- The function returns a 2D complex array of all zeroes except for a purely real 2D rect function. (Thus the imaginary component of the array will be all zeroes.)
- User input to the function is:

o image_x_width_cm – width of the image in the x-direction in cm.
o nx – number of pixels in the x direction.
o ny – number of pixels in the y direction.
o rect_ii_center_cm – ii coordinate of the center of the rect function. Please see Comment 1 below for additional details.
o rect_jj_center_cm – jj coordinate of the center of the rect function.
o rect_x_width_cm – width of the rect function in the x-direction, in cm.

o rect_y_width_cm – width of the rect function in the y-direction, in cm.

o 'even-width' or 'odd-width' – indicator of whether the width of the rect function is an even number of pixels or an odd number of pixels.

•The function should print messages indicating the following:

o The x- and y-axis widths of the image in cm.
o The x- and y-axis widths of the *implemented* rect function (a) in pixel widths and (b) in cm.

Comment 1: The coordinate ii should indicate x location in units of pixel width, with ii = 1 corresponding to the first column of pixels, ii = 2 corresponding to the second column of pixels, etc. Similarly, the coordinate jj should indicate j location in units of pixel width, with jj = 1 corresponding to the first row of pixels, etc.

Thus (rect_ii_center_cm, rect_jj_center_cm) = (1,1) would center the rect function on the first element of the array, which is presumably a corner on the displayed image.

Comment 2: The matlab function will need to wrap the rect to the other side of the image when the center of the rect is close to the x and/or y edges of the image. In other words, consider the image to be periodic in the x and y directions, as is standard for DFTs.

Comment 3: Since the user may enter rect_x_width_cm and rect_y_width_cm values that do not correspond exactly to an even or odd number of pixels, the function will need to find the odd or even number of pixels that best matches the user specified width.

Comment 4: This function will be needed below and in the next Matlab assignment.

3) Use your function to implement, 20-cm-wide, 128x128 pixel images with square 2D rect functions that are (nominally) 2 cm and 5 cm wide, have an odd number of pixels in each direction, and are centered at (ii,jj) = (1,1).

3a) Question B: What are the actual widths of the two rect functions, in pixels and in cm?

3b) Show pictures of the two rect images. (These pictures should have 1's near the 4 corners and zeroes elsewhere.)

3c) Take the DFTs of the two images.

- Question C: Are the imaginary components of the DFTs negligible compared to

  the real components? What are typical values for the real and imaginary

  components?

- Show images of the magnitudes (abs), phases, real components, and imaginary

  components of the DFTs.

- Question D: How do the magnitude images compare to the real images?

3d) Shift one of the rect functions away from (1,1) so that it is centered somewhere else in the image, e.g. maybe near the middle. Take the DFT. Show this image with the new rect function location and show the DFT. Question E: How does the imaginary component compare to real component? Is it still negligible?

4) Repeat (3) for the case of a 20-cm-wide square image with 32x32 pixels.

5a) Use cd to set the current folder to an empty folder, where you don't have any important files that you wouldn't want to inadvertently delete. (Be sure to use a relative, rather than full, sub-folder name, as described above.)

5b) Use fopen() to open output files, in the current folder, to which you can write the real components of the above images from part (3). Specify the fopen() permission option that will open a file, or create a new file, for writing; discarding existing contents, if any.

- Assign filenames rect_2cm, dft_2cm, rect_5cm, dft_5cm
- Assign file IDs rect_2cm_ID, dft_2cm_ID, rect_5cm_ID, dft_5cm_ID
  Check the current folder to verify that the files are present. Note that the file sizes are all zero.

  5c) Use fwrite() to write the real components of the above images, from parts (3), to the files you have opened.

• Check the sizes of the output files. The sizes may still be zero.

• Use fclose() to close the output files. Check the sizes again. They should be non-zero. Question G: What are the file sizes? Question H: Are the sizes about right given the number of elements in your images and given the default *precision* for the fwrite() function?

Note from the Matlab documentation that fwrite() creates binary files. Question I: What are binary files?

6) Download and install the program ImageJ from the NIH website. Use ImageJ to import and display the images that are stored in the files you created in part (5).

- File → Import → Raw
- Image → Adjust → Window/Level or Image → Adjust → Brightness/Contrast Question J: Are the ImageJ images similar to those you generated in Matlab? If not, how are they different? Include a screen-capture of the ImageJ images in your report.