1. Moire patterns are common in medical radiography and industrial images as shown in image_1.jpg and image_2.jpg. This problem studies how to reduce Moire patterns while properly preserving the salient features for diagnosis. Download the images image_1.jpg and image2_2.jpg. Each of these images is corrupted by a clearly visible Moire pattern.
    a. For each image, apply an N×N median filter (function: medfilt2). Adjust the window size N so that the Moire pattern is removed as much as possible while salient features are properly preserved. Report your choice of N, submit the filtered image, and comment on the quality of the filtered image.
    b. For each image, compute its Discrete Fourier Transform (DFT) (using functions: fft2 and fftshift) and submit an image showing the DFT magnitude (function: abs). A log display may be most appropriate. Clearly identify and label the frequency components that correspond to the Moire pattern.
    c. For each image, design a notch filter so that the frequency components for the Moire pattern are suppressed as much as possible while other frequency components are preserved. Apply your notch filter to the image's DFT and submit an image showing the filtered DFT magnitude. Display and submit the filtered image in the spatial domain (functions: ifftshift, ifft2, and real). Compare the quality of the result to that of the filtered image from part(a).
2. Download the images Blocks.jpg and Chess.png. Using an appropriate spatial or frequency domain filter, extract the edges in the images. For Blocks.jpg, the result should have only the edges of the block. For the Chess.png, the blocks of the chess board need to be extracted.
3. Plot the low/high decomposition and reconstruction filters for Daubechies 8-tap (db4) wavelet transform. You can obtain the Daubechies coefficients using the function "wfilters" in MATLAB or e.g. at http://wavelets.pybytes.com/family/db/. This is the wavelet transform you should use for all the remaining tasks.
4. Implement a 2D wavelet and inverse 2D wavelet transforms using 1D wavelet transforms. It should have the following interface:
   $y2d = wt2d(x2d, LoD, HiD, nlevels)$
   $x2d = iwt2d(y2d, LoR, HiR, nlevels)$,
   Where $x2d$ is the input image, LoD and LoR are the 1D low-pass approximation filters while HiD and HiR are the 1D high-pass detail filters for decomposition and reconstruction, respectively. The number of desired scales is specified by n levels. (The encoded signal $y2d$ should have the standard layout, having approximation at the top-left block and the detail coefficients around it). To test, run a 3-level decomposition and reconstruction of the selected test image (say barbara.png). Verify how well is the image reconstructed by computing the mean absolute error between the original and the reconstructed image.
5. Decompose and reconstruct the image using the approximation coefficients only. Perform this for 3 progressively larger levels. Visualize the original image and the three reconstructed images, side by side
6. Add Gaussian noise (μ= 0, σ= 10) to the original test image. Visualize both images (the original and its noisy version) side by side. Compute peak signal to noise ratio (PSNR) between the two images.
7. Perform Wavelet decomposition and thresholding the detail coefficients in the following way:
    a. Select a threshold $T = \sigma\sqrt{2\,log(n)}$, where n is the number of the detail coefficients and σ is an estimate of the noise level.
    b. Set all detail coefficients with their absolute value $< T$ to zero.
    c. Reconstruct the image. Compute the PSNR for different decomposition levels and σ values, i.e., for de-composition levels {1,2,3} and values of $\sigma = \{1,10,100\}$.