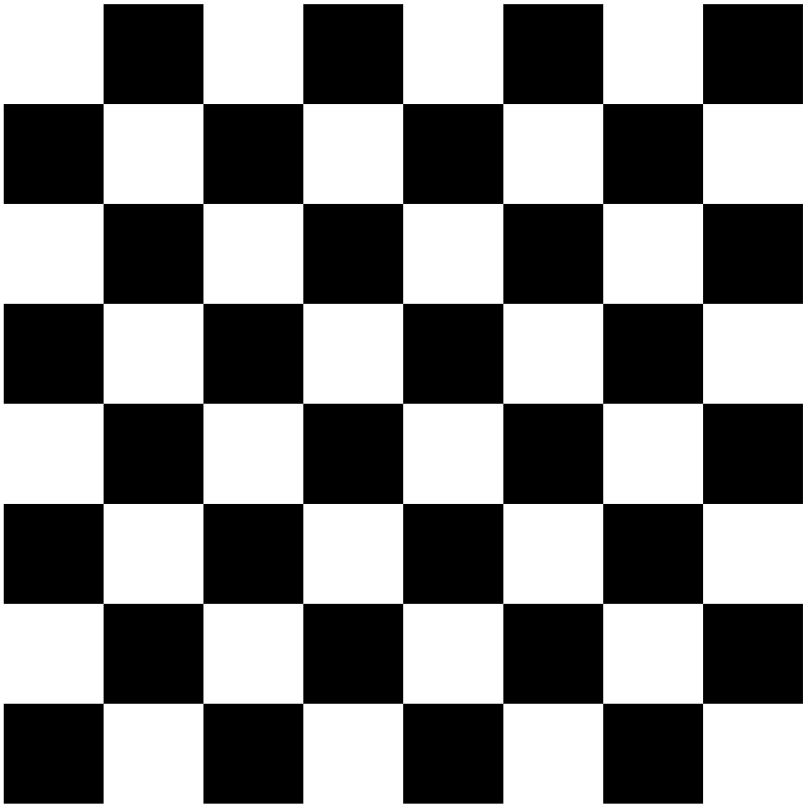
Module 1 Assignment

Questions 1 and 2: Working with Test Patterns

One way to get a feel for how spatial filters affect an image is to apply them to very simple test patterns. The following checkerboard pattern is composed of alternating 16-by-16 pixel squares.

```
A = ones(16);
B = zeros(16);
C = [A,B;B,A];
C = [C,C;C,C];
img = [C,C;C,C];
imshow(img)
```

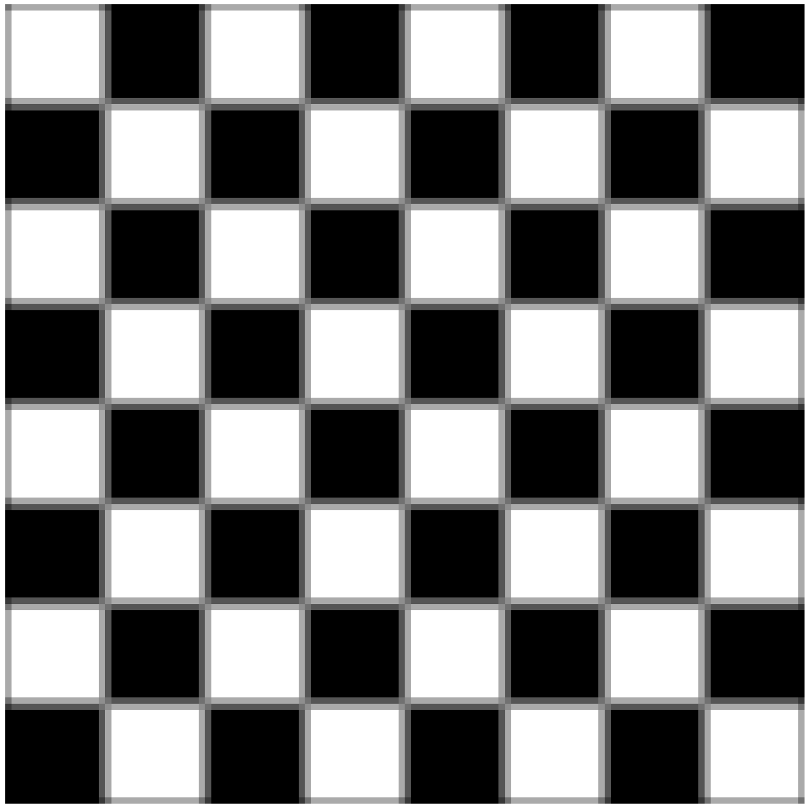


Get the result of applying 3x3 averaging filter

```
favg = fspecial("average",[3,3])
```

```
favg = 3×3
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
```

```
imgavg = imfilter(img,favg);
imshow(imgavg)
```

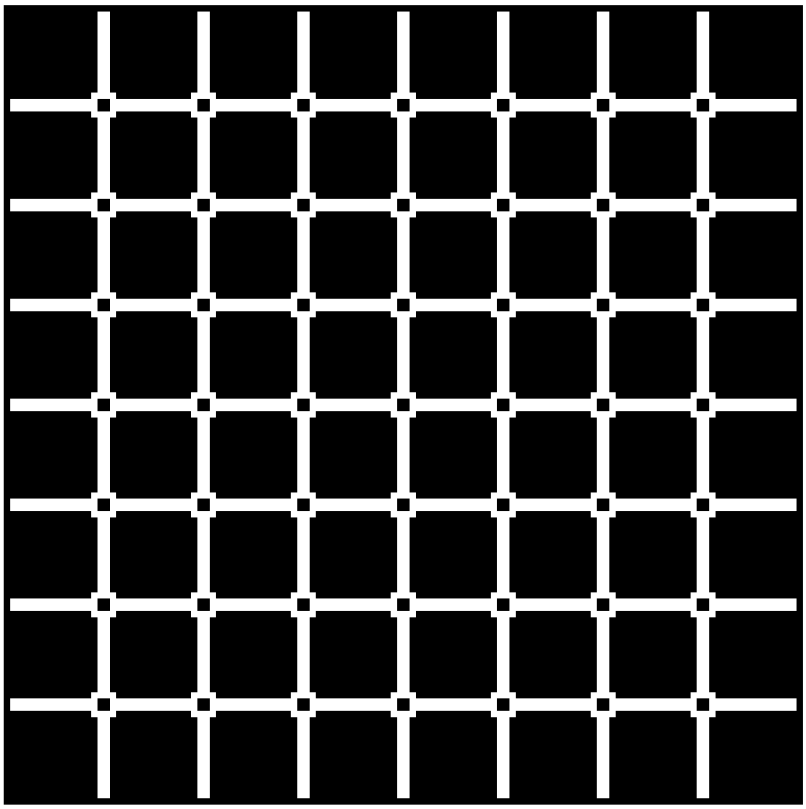


Get the result of applying Canny edge detection to the checkerboard image.

```
fcanny = edge(img,"canny")
```

```
fcanny = 128×128 logical array
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0···
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0
     ⋮
```

```
imshow(fcanny)
```



Question 3

Visualizing Gradients

Load the '*coins.png*' image included with MATLAB. Detect edges using the *edge* function with the Sobel method, and return the vertical and horizontal gradients.

Which image best represents the gradient highlighting the vertical edges in this image?

```
coin = imread("coins.png");
[coinEdge, threshOut, VerEdGrad,HEdGrad] = edge(coin,"sobel");
imshow(VerEdGrad)
```

Question 4

Detecting Edges in Coins

Load the *coins1.jpg* image and resize it to a resolution of 600 by 800 pixels. Apply a Gaussian filter with a standard deviation of 0.5. Which of these images shows the edges detected by the *edge* function with the Canny method?

```
coin1 = imread("coins1.jpg");
img = imresize(coin1,[600,800]);
imshow(img)
```

```
imgfilt = imgaussfilt(img,0.5);
imgcanny = edge(imgfilt,"canny");
imshow(imgcanny)
```

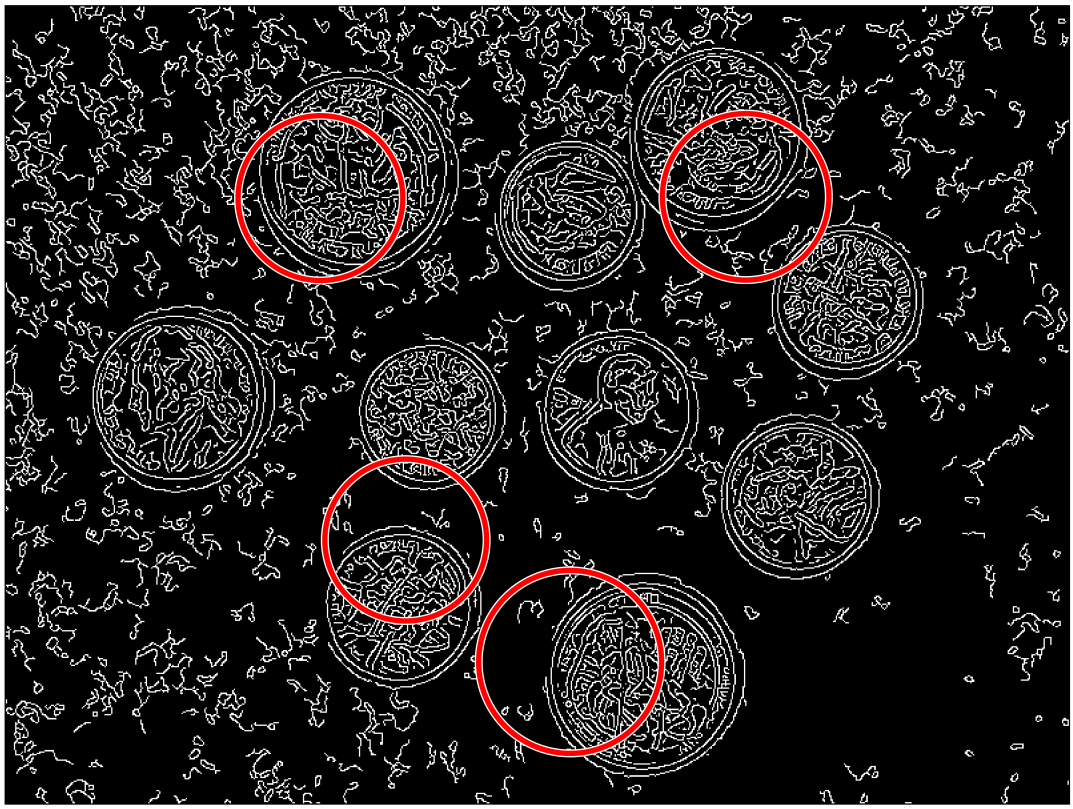
Questions 5 and 6: Counting Coins

The next two questions involve using the *imfindcircles* and *viscircles* functions. For each question, you will need to load the image, resize it, and apply the specified Gaussian blur with the *imgaussfilt* function.

Counting Coins 1

Load the *coins3.jpg* image and rescale the image to 600 by 800 pixels. Blur the image using a Gaussian blur with a standard deviation (*sigma*) of 1. Use the *imfindcircles* function to find only the four largest coins. What is the lower value of the radius search to eliminate the smaller coins?

```
coin3 = imread("coins3.jpg");
img = imresize(coin3,[600,800]);
imgfilt = imgaussfilt(img,1);
[centers,radii] = imfindcircles(imgfilt,[56,100]);
viscircles(centers,radii)
```



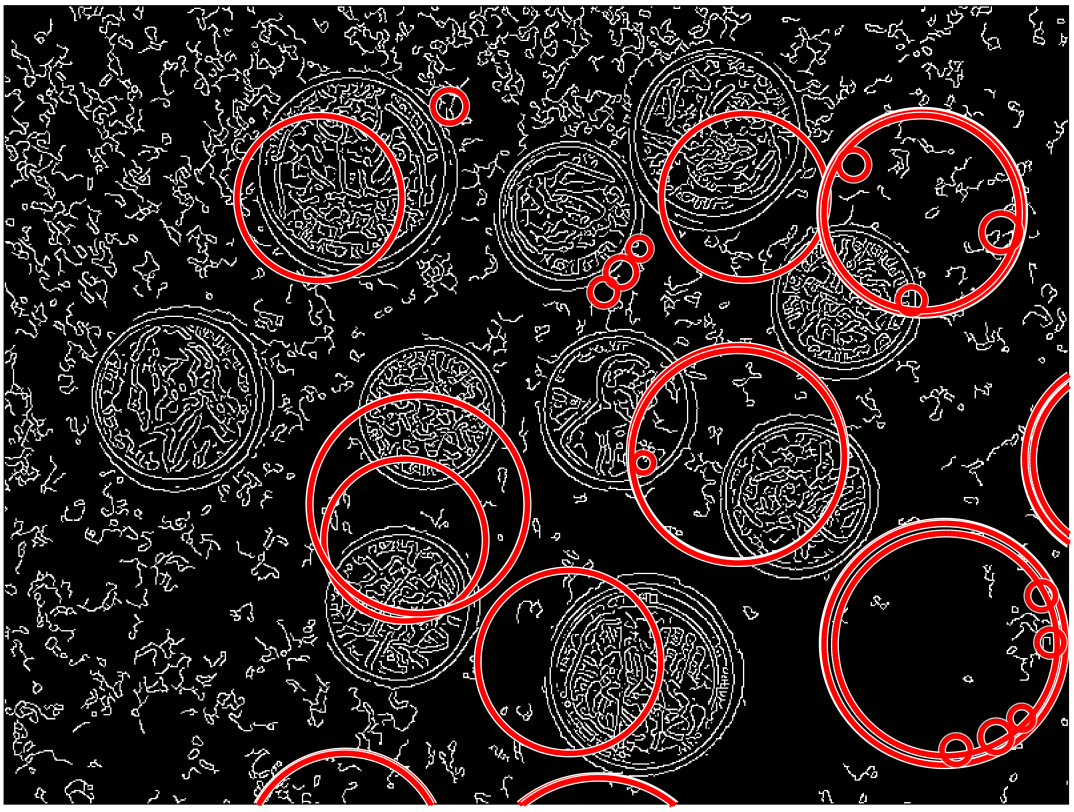
```
ans =
  Group with properties:

    Children: [2×1 Line]
     Visible: on
     HitTest: on

  Show all properties
```

Question 6

Counting Coins 2

Load the *coins2.jpg* image and rescale the image to 900 by 1200 pixels. Blur the image using a Gaussian blur with a standard deviation (*sigma*) of 1. Use the *imfindcircles* function to count the number of coins with radius less than 100.

```
coin2 = imread("coins2.jpg");
img = imresize(coin2,[900,1200]);
imgfilter = imgaussfilt(img,1);
[centers,radii] = imfindcircles(imgfilter,[6,100]);
viscircles(centers,radii)
```



```
ans =
  Group with properties:

    Children: [2×1 Line]
     Visible: on
     HitTest: on

  Show all properties
```