Exercise 1

Part A at bottom of script (function declaration)

```
%Part B
A = 0.8;
s = 8000;
tyb_orig = data;
tyb_reverb = reverb_conv(tyb_orig, s, A);
sound(tyb_reverb, fs);
tyb_orig = data;
pause(6);

%Part D
A = 0.3;
fm = 10 / fs;
tyb_tremelo = tremelo(tyb_orig,fm,A);
soundsc(tyb_tremelo,fs);
pause(6);

%Part E
A = 1;
fm = 1 / length(tyb_orig);
tyb_faded = tremelo(tyb_orig,fm,A);
soundsc(tyb_faded,fs);
pause(6);
```

Part D explanation

I hear the original sound at the beginning, then it fades till it is silentin the middle, thenreturns to the original sound at the end

```
N = zeros(123480,1);
tyb_faded_N = [N; tyb_faded];
soundsc(tyb_faded_N, fs);
pause(7);

tyb_orig_N = [N; tyb_orig];
tyb_N_faded = tremelo(tyb_orig_N,fm,A);
soundsc(tyb_N_faded,fs);
```

Part F explanation

These two signals do not sound the same as the fade wave caused by the tremelo effect starts at its minimum (original volume) and gradually intensifies it peaks in the middle of the original wave (no volume). By applying the fade first, it makes the middle of the clip fade away, whereas applying the shift first makes the beginning and end of the fade away.

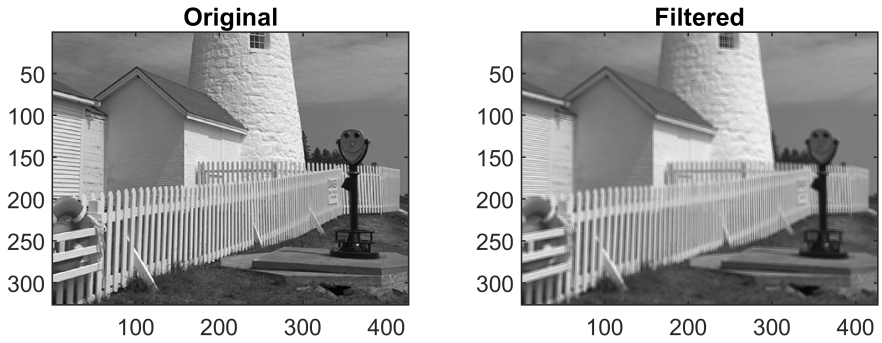
Exercise 2

```
%Part A
lighthouse;
wa = zeros(3) + (1/3);
l = filter2(wa, lighthouse);

subplot (1,2,1);
imagesc(lighthouse);
axis image;
title("Original")

subplot (1,2,2);
imagesc(l);
axis image;
title("Filtered")
colormap(gray);
```

Part A explanation

The filtered image seems to have a bit more general blurring. This is because the filter convolves the image with the kernel to make the difference in neighboring pixels much more subtle than in the original image.

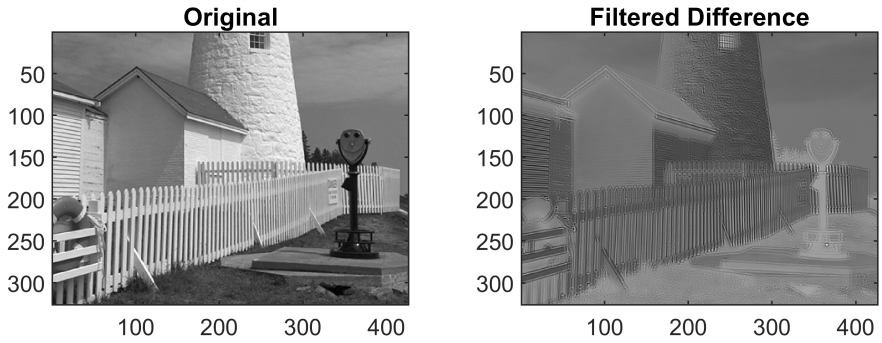
```
%Part B
wb = zeros(3) + (1/3);
wb(2,2) = -8/9;
l = filter2(wb, lighthouse);

figure(1)
subplot (1,2,1);
imagesc(lighthouse);
axis image;
title("Original")
subplot (1,2,2);
imagesc(l);
axis image;
title("Filtered")
colormap(gray);
```



Part B explanation

The filtered image seems to have slightly more blurring than the first filter with the color differences, but more sharpness with the edges of shapes (ex the fence and the ground are more emphazed). This seems to work because the filter keep center pixel "strong" in its state while "averaging" the negihboring pixels to its tone.

```
%Part C
l_out = image_unsharp_masking(lighthouse);

figure(2)
subplot (1,2,1);
imagesc(lighthouse);
axis image;
title("Original")
subplot (1,2,2);
imagesc(l_out);
axis image;
title("Filtered Difference")
colormap(gray);
```

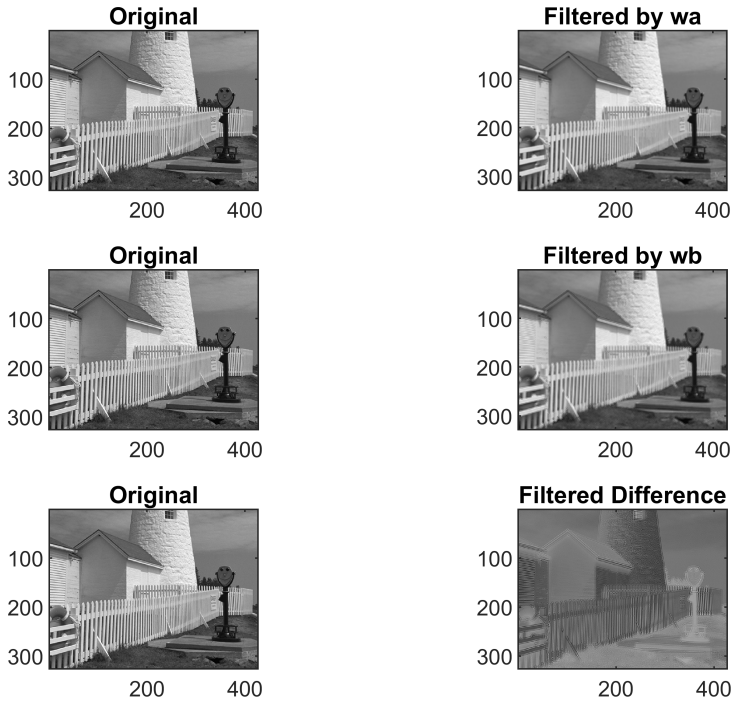


Part C explanation

It appears that this operation nearly inverts the image, while also blurring it

```
%Part D
lighthouse_filtered_by_wa = conv2(wa, lighthouse);

figure(3)
subplot (3,2,1);
imagesc(lighthouse);
axis image;
title("Original")
subplot (3,2,2);
imagesc(lighthouse_filtered_by_wa);
axis image;
title("Filtered by wa")


lighthouse_filtered_by_wb= conv2(wb, lighthouse);

subplot (3,2,3);
imagesc(lighthouse);
axis image;
title("Original")
subplot (3,2,4);
imagesc(lighthouse_filtered_by_wb);
axis image;
title("Filtered by wb")
```

```
l_out = image_unsharp_masking(lighthouse);

subplot (3,2,5);
imagesc(lighthouse);
axis image;
title("Original")
subplot (3,2,6);
imagesc(l_out);
axis image;
title("Filtered Difference")
colormap(gray);
```



FUNCTION DECLARATIONS

Exercise 1

Part A

```
function y = reverb_own(x, s, A)
    %Less efficient with zeros spread across the vectors
   d = zeros(s, 1);
   x_delayed = [d; x];
   x_longer = [x; d];
   y = x_longer + A*x_delayed;
end
function y = reverb_conv(x, s, A)
%Computationally tedious as it must work through 2 functions: conv and conv2
```

```
    h = [1; zeros(s,1);A];
    y = conv(x, h);
end
function y = reverb_filter(x, s, A)
    %Hard to say how the computational efficiency differs from the other
    %functions because source code is not available
     h = [1; zeros(s,1);A];
    y = filter(h,1,x);
end
```

Part C

```
function y = tremelo(x,fm,A)
    %Subtle changes to the sound, makes the sound "tremble"
    for i = 1 : size(x,1)
        y(i) = x(i) + A*cos(2*pi*fm*i)*x(i);
    end
    y = y';
end
```

Exercise 2

Part C

```
function im_out = image_unsharp_masking(im_in)
    wb = zeros(3) + (1/3);
    wb(2,2) = -8/9;
    im_out = filter2(wb, im_in);
    im_out = im_in - im_out;
end
```