Assignment Two

For this assignment, we were to produce a color image by extracting three color channel images and aligning them to form a single colored image. As seen with the previous assignment, it was important to implement an efficient algorithm that utilizes a recursive pyramid of resized images to ensure the large images processed at a quicker rate.

This was accomplished by utilizing the sum of squared differences to compute the best alignment match. By computing this value, we were able to, in effect, rate each comparison and ultimately pick the best match. This was successful, as evidenced in the photos below that are properly aligned.

To implement this, my functions are listed below:

```matlab
function start
image = imread('00084v.jpg');
dim = size(image);
length = dim(1);
width = dim(2);
third = length/3;
B = image(1:third, :);
G = image(third:third*2, :);
R = image(third*2 : length, :);
B = im2double(B);
G = im2double(G);
R = im2double(R);
second(3, R, G, B);

function [bx, by, rx, ry] = second(level, R, G, B)
% recursively call function
if level ~= 0
    B = imresize(B, .5);
    R = imresize(R, .5);
    G = imresize(G, .5);
    [bx, by, rx, ry] = second(level -1, R, G, B);
    B = circshift(B, [bx*2, by*2]);
    R = circshift(R, [rx*2, ry*2]);
end
% Use circshift to find the best match up of R and B with G
x = 0;
y = 0;
best = intmax('int32');
for i = -16 : 16
    for j = -16 : 16
        % find the sum of squared differences between the images
```
shifted = circshift(R,[i j]);
match = sum(sum((shifted-G).^2));

    if match < best
        best = match;
        bestimage = shifted;
        rx = i;
        ry = j;
    end
end

end

combination(:,:,1) = bestimage(:,:,1);
combination(:,:,2) = G(:,:,1);

x = 0;
y = 0;
best = intmax('int32');
for i = -16 : 16
    for j = -16 : 16

        % find the sum of squared differences between the images
        shifted = circshift(B,[i j]);
        match = sum(sum((shifted-G).^2));
        if match < best
            best = match;
            bestimage = shifted;
            bx = i;
            by = j;
        end
    end
end

combination(:,:,3) = bestimage(:,:,3);
imshow(combination)

Results:
The results were quite nice for the smaller images. Running the program for those was quick and efficient and produced high quality color representations. The issue arises when trying to run this on a larger sized image. As evidenced above, the process still works properly, however, the time complexity is still much too high. In order to make this run more efficiently, it would be wise to look into more strategies discussed, like calculating the normalized correlation. Optimizing this algorithm would be very valuable. By increasing the number of recursive image resizes, we could search a much smaller image, which might be an improvement. This could help provide a less costly means to find the proper alignment.