Practical 1

a) Display and Access Pixel Intensity Values in Grayscale Image.

MATLAB Code:

```
img = imread('cameraman.tif');
imshow(img);
title('Grayscale Image');

intensity = img(50, 60);
fprintf('Pixel intensity at (50,60) is: %d\n', intensity);
```

OUTPUT

```
>> practical1a
Pixel intensity at (50,60) is: 179
```

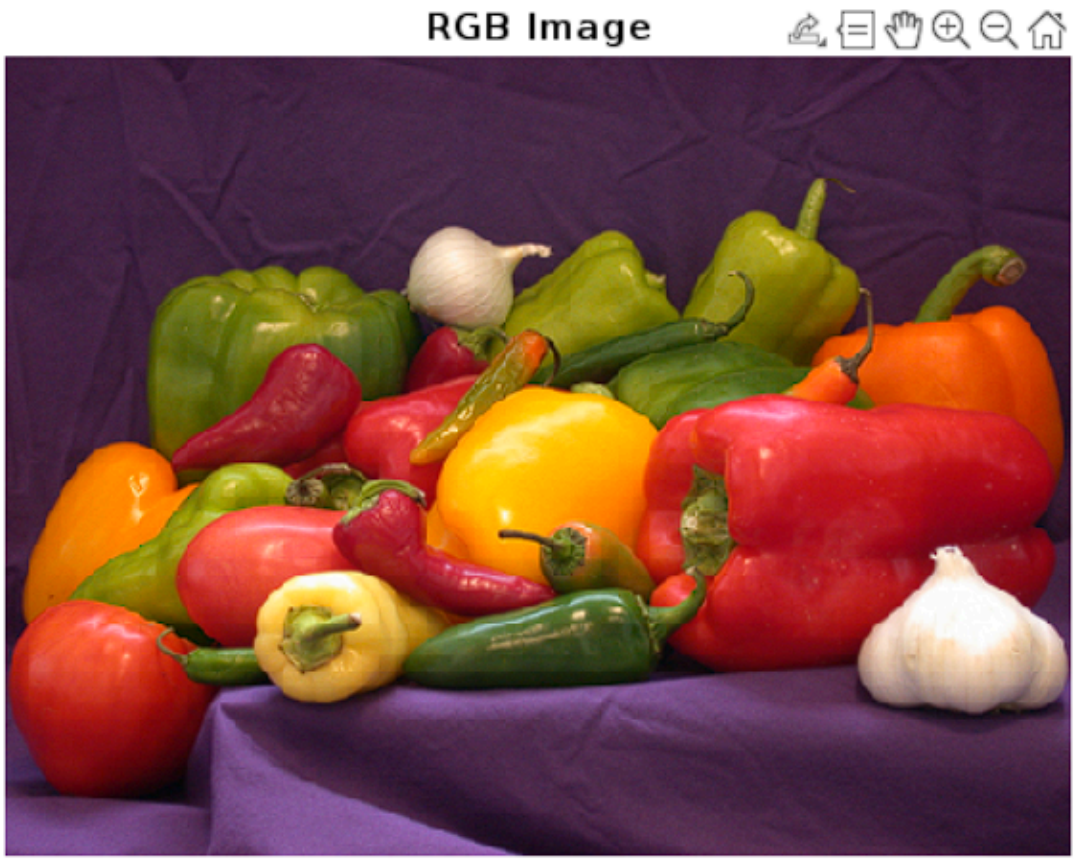
b) Display RGB Image and Show Its Dimensions.

MATLAB Code:

```
img = imread('peppers.png');
imshow(img);
title('RGB Image');

[rows, cols, channels] = size(img);
fprintf('Dimensions: %d rows x %d cols x %d channels\n', rows, cols, channels);
```

OUTPUT

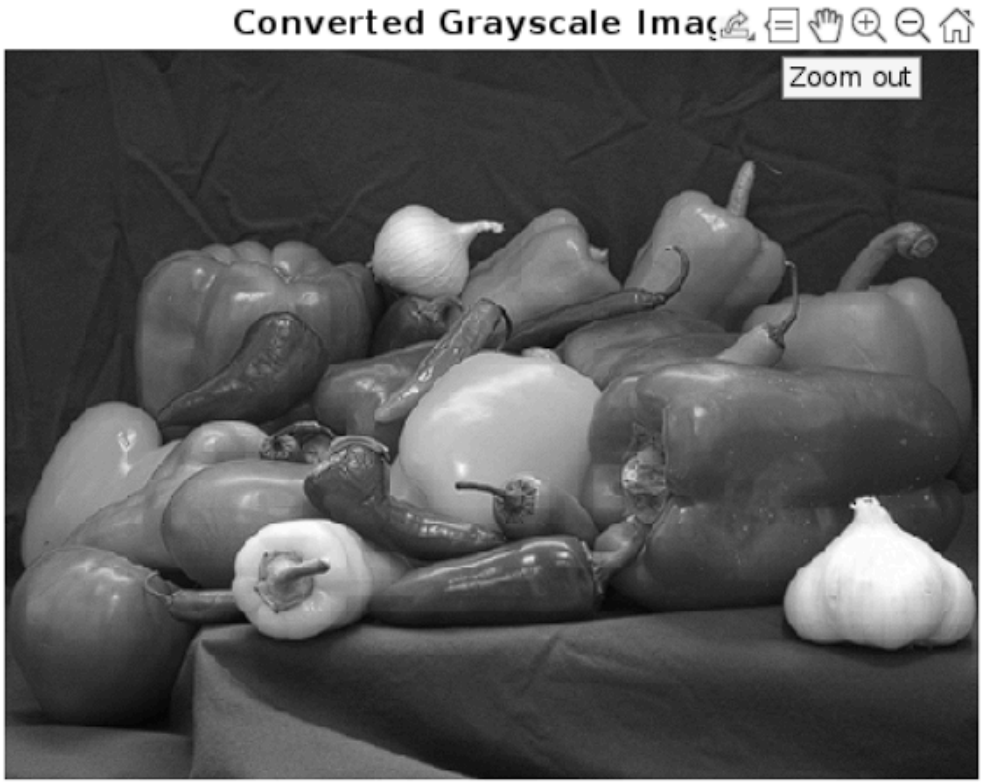
```
>> practical1b
Dimensions: 384 rows x 512 cols x 3 channels
```



c) Convert RGB Image into Grayscale Image

```
rgb = imread('peppers.png');
gray = rgb2gray(rgb);
imshow(gray);
title('Converted Grayscale Image');
```

OUTPUT



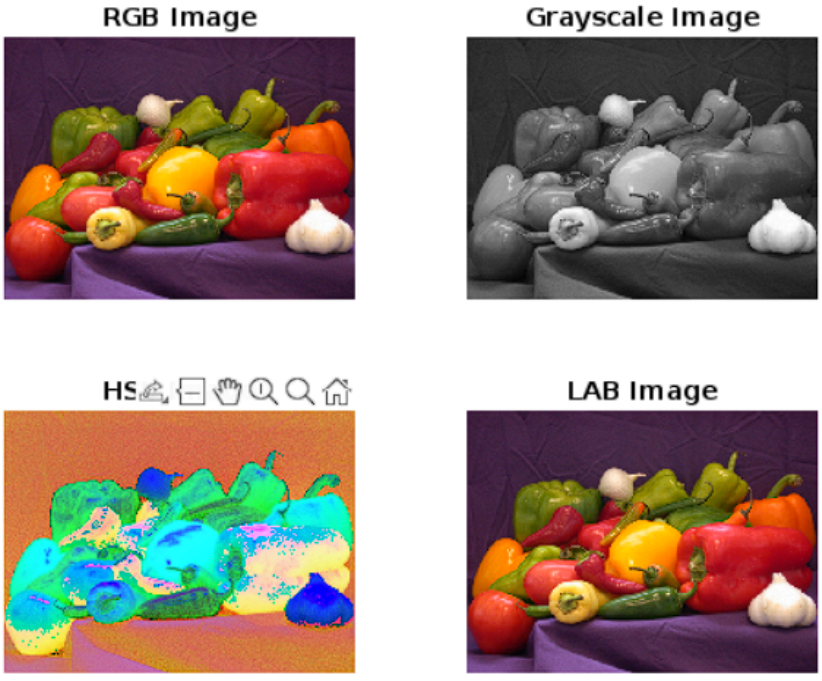
Practical 2

a) Convert RGB to Different Color Spaces and Display with Subplots

MATLAB Code:

```
rgb = imread('peppers.png');

lab = rgb2lab(rgb);

hsv = rgb2hsv(rgb);

gray = rgb2gray(rgb);

subplot(2,2,1); imshow(rgb); title('RGB Image');

subplot(2,2,2); imshow(gray); title('Grayscale Image');

subplot(2,2,3); imshow(hsv); title('HSV Image');

subplot(2,2,4); imshow(lab2rgb(lab)); title('LAB Image');
```

OUTPUT

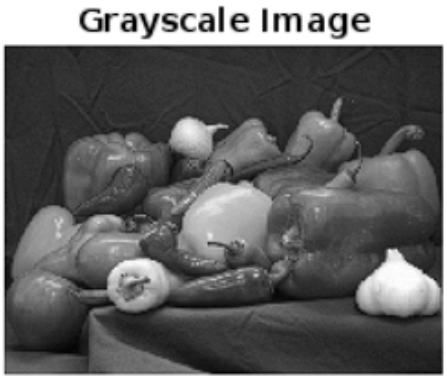


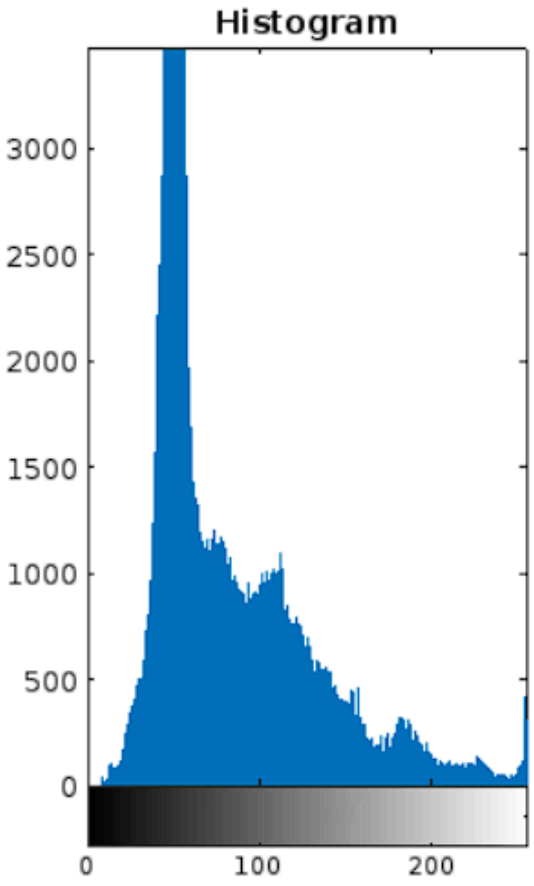
b) Histogram Processing of Grayscale Image

MATLAB Code:

```
img = rgb2gray(imread('peppers.png'));

figure;

subplot(1,2,1); imshow(img); title('Grayscale Image');

subplot(1,2,2); imhist(img); title('Histogram');
```

OUTPUT





Practical 3

a) Display Image Metadata

MATLAB Code:

```
info = imfinfo('peppers.png');

disp(info);
```

OUTPUT

```
>> practical3a
                Filename: '/MATLAB/toolbox/matlab/matlab_images/png/peppers.png'
             FileModDate: '13-Jul-2023 20:26:13'
                FileSize: 287677
                  Format: 'png'
           FormatVersion: []
                   Width: 512
                  Height: 384
                BitDepth: 24
               ColorType: 'truecolor'
         FormatSignature: [137 80 78 71 13 10 26 10]
                Colormap: []
               Histogram: []
           InterlaceType: 'none'
            Transparency: 'none'
   SimpleTransparencyData: []
         BackgroundColor: []
         RenderingIntent: []
          Chromaticities: []
                   Gamma: []
             XResolution: []
             YResolution: []
          ResolutionUnit: []
                 XOffset: []
                 YOffset: []
              OffsetUnit: []
         SignificantBits: []
            ImageModTime: '16 Jul 2002 16:46:41 +0000'
                   Title: []
                  Author: []
             Description: 'Zesty peppers'
               Copyright: 'Copyright The MathWorks, Inc.'
            CreationTime: []
                Software: []
              Disclaimer: []
                 Warning: []
                  Source: []
                 Comment: []
               OtherText: []
       AutoOrientedWidth: 512
      AutoOrientedHeight: 384

>>
```

b) Perform Uniform Quantization

MATLAB Code:

```
img = rgb2gray(imread('peppers.png'));

levels = 4; % e.g., 4 levels: 0, 85, 170, 255

quantized_img = uint8(floor(double(img)/256 * levels) * (256/levels));

imshow(quantized_img);

title('Uniform Quantized Image');
```

OUTPUT

Practical 4

a) Image Arithmetic using Switch Case

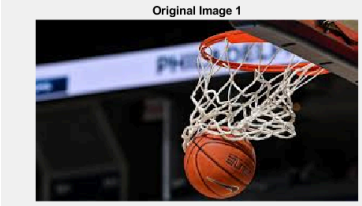
MATLAB Code:

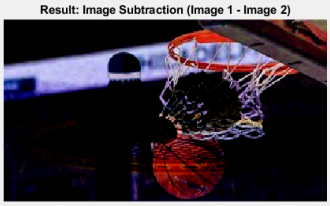
```
clc;
close all;
clear all;

% --- 1. Read Input Images ---
% directory as this script, or provide the full path.
img1 = imread('basketball.jpeg');
img2 = imread('badminton.jpg');

% --- 2. Pre-processing: Ensure Images are the Same Size ---
[rows, cols, ~] = size(img1);
img2_resized = imresize(img2, [rows, cols]);

% --- 3. User Menu and Input ---
fprintf('Image Arithmetic Operations Menu:\n');
fprintf('1. Add two images\n');
fprintf('2. Subtract two images\n');
fprintf('3. Multiply two images\n');
fprintf('4. Divide two images\n');
fprintf('5. Add a constant value (50) to an image\n');
fprintf('6. Subtract a constant value (50) from an image\n');
fprintf('7. Multiply an image by a constant value (1.5)\n');
fprintf('8. Divide an image by a constant value (1.5)\n');

choice = input('Enter your choice (1-8): ');

% --- 4. Perform Operation using Switch Case ---
switch choice
   case 1
      result_img = imadd(img1, img2_resized);
      operation_title = 'Image Addition (Image 1 + Image 2)';

   case 2
      result_img = imsubtract(img1, img2_resized);
      operation_title = 'Image Subtraction (Image 1 - Image 2)';

   case 3
      result_img = immultiply(img1, img2_resized);
```

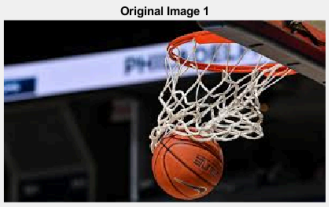
```
        operation_title = 'Image Multiplication (Image 1 * Image 2)';

    case 4
        result_img = imdivide(img1, img2_resized);
        operation_title = 'Image Division (Image 1 / Image 2)';

    case 5
        constant_val = 50; % Predefined constant
        % The constant must be a double, so we do not cast to uint8
        result_img = imadd(img1, constant_val);
        operation_title = ['Constant Addition (Image 1 + ' num2str(constant_val) ')'];

    case 6
        constant_val = 50; % Predefined constant
        % The constant must be a double, so we do not cast to uint8
        result_img = imsubtract(img1, constant_val);
        operation_title = ['Constant Subtraction (Image 1 - ' num2str(constant_val) ')'];

    case 7
        constant_val = 1.5; % Predefined constant
        result_img = immultiply(img1, constant_val);
        operation_title = ['Constant Multiplication (Image 1 * ' num2str(constant_val) ')'];

    case 8
        constant_val = 1.5; % Predefined constant
        result_img = imdivide(img1, constant_val);
        operation_title = ['Constant Division (Image 1 / ' num2str(constant_val) ')'];
    otherwise
        error('Invalid choice. Please run the script again and select a number between 1 and 8.');
end

% --- 5. Display Results ---
figure('Name', 'Image Arithmetic Results', 'NumberTitle', 'off');

subplot(1, 3, 1);
imshow(img1);
title('Original Image 1');

subplot(1, 3, 2);
imshow(img2_resized);
title('Original Image 2 (Resized)');

subplot(1, 3, 3);
imshow(result_img);
```

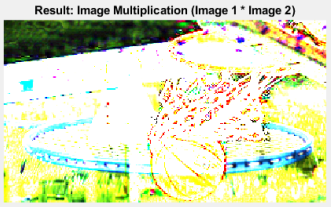
```
title(['Result: ' operation_title]);
```

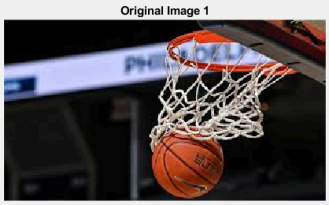
OUTPUT

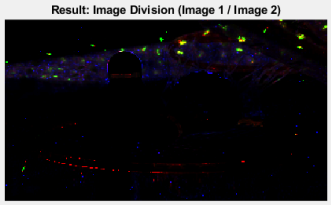


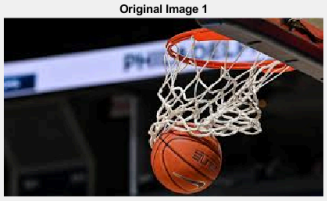


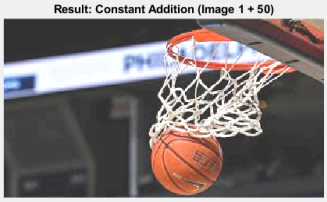


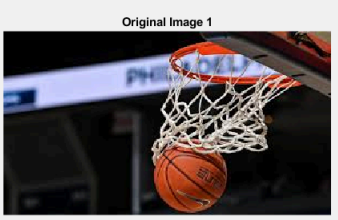


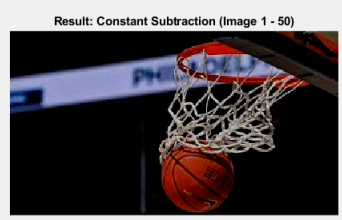












Result Findings:

1. Image Addition: Blends the two images, resulting in a much brighter, combined image.

2. Image Subtraction: Highlights the differences. Similar areas become dark, different areas stay bright.

3. Image Multiplication: Makes the image very dark. It's mainly used for masking effects.

4. Image Division: Creates a high-contrast, often noisy-looking image that also highlights differences.

5. Constant Addition (+50): Makes the image uniformly brighter.

6. Constant Subtraction (-50): Makes the image uniformly darker.

7. Constant Multiplication (x1.5): Increases the image's brightness and contrast.

8. Constant Division (/1.5): Decreases the image's brightness.

b) Bitwise Operations using Switch Case

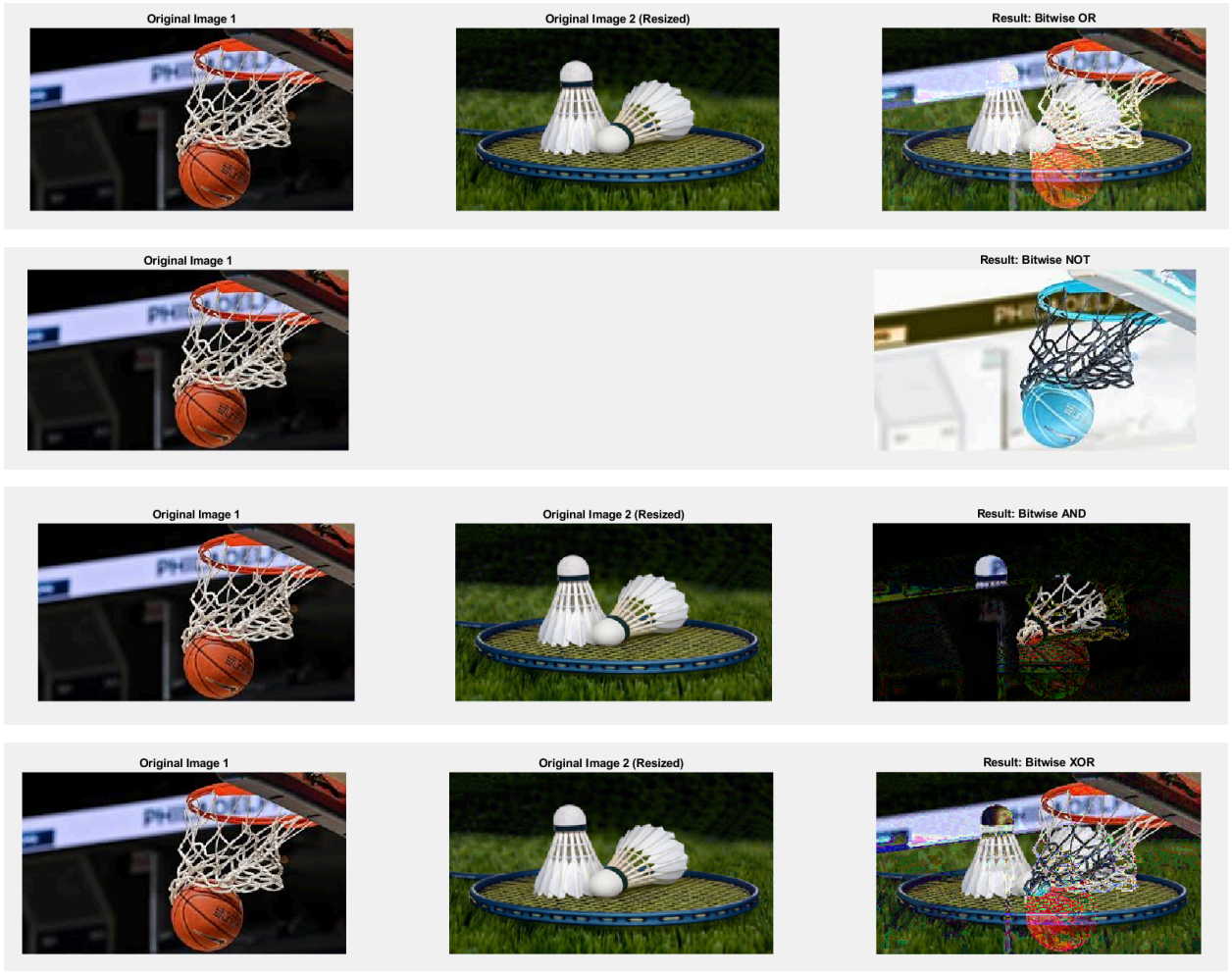
MATLAB Code:

```
clc;
close all;
clear all;

img1 = imread('basketball.jpg');
img2 = imread('badminton.jpg');

[rows, cols, ~] = size(img1);
```

```
img2_resized = imresize(img2, [rows, cols]);

fprintf('Image Bitwise Operations Menu:\n');
fprintf('1. Bitwise AND\n');
fprintf('2. Bitwise OR\n');
fprintf('3. Bitwise NOT (on Image 1)\n');
fprintf('4. Bitwise XOR\n');

choice = input('Enter your choice (1-4): ');

switch choice
    case 1
        result_img = bitand(img1, img2_resized);
        operation_title = 'Bitwise AND';

    case 2
        result_img = bitor(img1, img2_resized);
        operation_title = 'Bitwise OR';

    case 3
        result_img = bitcmp(img1);
        operation_title = 'Bitwise NOT';

    case 4
        result_img = bitxor(img1, img2_resized);
        operation_title = 'Bitwise XOR';

    otherwise
        error('Invalid choice. Please run the script again and select a number between 1 and 4.');
end

figure('Name', 'Image Bitwise Results', 'NumberTitle', 'off');

subplot(1, 3, 1);
imshow(img1);
title('Original Image 1');

subplot(1, 3, 2);
if choice ~= 3
    imshow(img2_resized);
    title('Original Image 2 (Resized)');
else
    axis off;
    title(' ');
```

```
end

subplot(1, 3, 3);
imshow(result_img);
title(['Result: ' operation_title]);
```

OUTPUT



RESULT FINDINGS:

1. AND: Creates a dark image, keeping only the features that are bright in both original images. It's useful for masking.
2. OR: Creates a bright, combined image, showing all the features that were bright in either of the original images.
3. NOT: Inverts the image, creating a photographic negative. Bright areas become dark, dark areas become bright, and colors are flipped.
4. XOR: Highlights only the exact differences between the two images. Areas that are the same in both images become black.

c) RGB Channel Separation

MATLAB Code:

```
clc;
close all;
clear all;

original_img = imread('basketball.jpg');

red_channel = original_img;
red_channel(:,:,2) = 0;
red_channel(:,:,3) = 0;

green_channel = original_img;
green_channel(:,:,1) = 0;
green_channel(:,:,3) = 0;

blue_channel = original_img;
blue_channel(:,:,1) = 0;
blue_channel(:,:,2) = 0;

figure('Name', 'RGB Channel Separation Results', 'NumberTitle', 'off');

subplot(2, 2, 1);
imshow(original_img);
title('Original RGB Image');

subplot(2, 2, 2);
imshow(red_channel);
title('Red Channel');

subplot(2, 2, 3);
imshow(green_channel);
title('Green Channel');

subplot(2, 2, 4);
imshow(blue_channel);
title('Blue Channel');
```

OUTPUT