```
% Problem 5: Image Interpolation: Nearest Neighbor and Bilinear Interpolation
clear all;

input_image = imread('einstein.tif'); % reading the original image. This image is 8↙
bits/pixel gray-scale
% image.
size_of_image = size(input_image); % It will simply calculate the size of the image.

downsampling_factor = 3; % we are just defining the downsampling factor. It is as per↙
given in the problem
% statement.

% Downsampling the image in horizontal and vertical directions
downsampled_image_d3 = input_image((downsampling_factor+1)/2:downsampling_factor:end,↙
...
    (downsampling_factor+1)/2:downsampling_factor:end); % it will downsample the image↙
by a factor of D in
% horizontal and vertical directions, where the top-left sample should be at the↙
location of ((D+1)/2, (D+1)/2)
% in the original image.

% upsampling
upsampled_image_d3 = zeros(size_of_image); % It will simply upsample the image.↙
Upsampling of the downsampled
% image to have exactly the size of the original image by filling zeros in the missing↙
pixels.

p = 1; % initializing the value of p.
for i = (downsampling_factor+1)/2:downsampling_factor:size_of_image % for loop as per↙
the given condition.
    q = 1; % initializing the value of q.
    for j = (downsampling_factor+1)/2:downsampling_factor:size_of_image
    upsampled_image_d3(i,j) = downsampled_image_d3(p,q); % for loop as per the given↙
condition.
        q = q + 1; % incrementing the value of q.
    end % end for loop.
    p = p + 1; % incrementing the value of q.
end % end for loop.


% Convolution with block shape for D=3
block_filter_d3 = [1 1 1]; % creating the block filter of shape and size D.
block_conv_image_d3 = upsampled_image_d3; % the block convolutional filter shold be of↙
the same size as of the
% upsampled image.
for i = 1:size_of_image % for loop condition
block_conv_image_d3(i,:) = conv(block_conv_image_d3(i,:), block_filter_d3, 'same'); %↙
Here, the convolutions
% of the rows of the matrix. convolution operation for rows.
end % for loop ends
```

```
for j = 1:size_of_image % for loop condition
block_conv_image_d3(:,j) = conv(block_conv_image_d3(:,j), block_filter_d3.','same'); %↙
Here, the convolutions
% of the rows of the matrix. convolution operation for columns.
end % for loop ends

% Convolution with triangle shape for D=3
triangle_filter_d3 = (1/3) * [1 2 3 2 1]; % defining the filter as per the given↙
problem.
triangle_conv_image_d3 = upsampled_image_d3; % the block convolutional filter shold be↙
of the same size as of the
% upsampled image.
for i = 1:size_of_image % for loop condition
    triangle_conv_image_d3(i,:) = conv(triangle_conv_image_d3(i,:),↙
triangle_filter_d3,'same'); % Here, the
    % convolutions of the rows of the matrix. convolution operation for rows.
end % for loop ends
for j = 1:size_of_image % for loop condition
    triangle_conv_image_d3(:,j) = conv(triangle_conv_image_d3(:,j),↙
triangle_filter_d3.', 'same'); % Here, the
    % convolutions of the rows of the matrix. convolution operation for columns.
end % for loop ends



% Figure plotting for D=3
figure; % figure creates figure graphics objects. figure objects are the individual↙
windows on the screen
% in which MATLAB displays graphical output.
subplot(2,2,1); % subplot(m, n, p) divides the current figure into an m-by-n grid and↙
creates axes in the
% position specified by p. Here, m=n=2, p=1.
imshow(input_image); % It will display the gray-scale image in the figure.
title('Original image'); % It will add the specified title for the current plot.
subplot(2,2,2); % subplot(m, n, p) divides the current figure into an m-by-n grid and↙
creates axes in the
% position specified by p. Here, m=n=p=2.
imshow(uint8(upsampled_image_d3)); % It will display the gray-scale image in the figure↙
and
% it will convert each and every pixel value of the input image into the range of 0 to↙
255.
title('Upsampled image (D=3)'); % It will add the specified title for the current plot.
subplot(2,2,3); % subplot(m, n, p) divides the current figure into an m-by-n grid and↙
creates axes in the
% position specified by p. Here, m=n=2, p=3.
imshow(uint8(block_conv_image_d3)); % It will display the gray-scale image in the↙
figure and
% it will convert each and every pixel value of the input image into the range of 0 to↙
255.
title('Interpolated image using Nearest Neighbour (D=3)'); % It will add the specified↙
```

```
title for the current plot.
subplot(2,2,4); % subplot(m, n, p) divides the current figure into an m-by-n grid and↙
creates axes in the
% position specified by p. Here, m=n=2, p=4.
imshow(uint8(triangle_conv_image_d3)); % It will display the gray-scale image in the↙
figure and
% it will convert each and every pixel value of the input image into the range of 0 to↙
255.
title('Image using Bilinear interpolation (D=3)'); % It will add the specified title↙
for the current plot.



% D = 7
downsampling_factor = 7; % we are just defining the downsampling factor. It is as per↙
given in the problem
% statement.

%Downsampling the image in horizontal and vertical directions
downsampled_image_d7 = input_image((downsampling_factor+1)/2:downsampling_factor:end,↙
...
    (downsampling_factor+1)/2:downsampling_factor:end); % it will downsample the image↙
by a factor of D in
% horizontal and vertical directions, where the top-left sample should be at the↙
location of ((D+1)/2, (D+1)/2)
% in the original image.

%upsampling
upsampled_image_d7 = zeros(size_of_image); % It will simply upsample the image.↙
Upsampling of the downsampled
% image to have exactly the size of the original image by filling zeros in
% the missing pixels.

p = 1; % initializing the value of p.
for i = (downsampling_factor+1)/2:downsampling_factor:size_of_image % for loop as per↙
the given condition.
    q = 1; % initializing the value of q.
    for j = (downsampling_factor+1)/2:downsampling_factor:size_of_image
    upsampled_image_d7(i,j) = downsampled_image_d7(p,q); % for loop as per the given↙
condition.
        q = q + 1; % incrementing the value of q.
    end % end for loop.
    p = p + 1; % incrementing the value of q.
end % end for loop.


% Convolution with block shape for D=7
block_filter_d7 = [1 1 1 1 1 1 1];  % defining the filter as per the given problem.
block_convoluted_image_d7 = upsampled_image_d7; % the block convolutional filter shold↙
be of the same size as
```

```
% of the upsampled image.
for i = 1:size_of_image % for loop condition
block_convoluted_image_d7(i,:) = conv(block_convoluted_image_d7(i,:), block_filter_d7,↙
'same'); % Here, the
    % convolutions of the rows of the matrix. convolution operation for rows.
end % for loop ends
for j = 1:size_of_image % for loop condition
block_convoluted_image_d7(:,j) = conv(block_convoluted_image_d7(:,j),↙
block_filter_d7.','same'); % Here, the
    % convolutions of the rows of the matrix. convolution operation for columns.
end % for loop ends

% Convolution with triangle shape for D=7
triangle_filter_d7 = (1/7) * [1 2 3 4 5 6 7 5 4 3 2 1]; % defining the filter as per↙
the given problem.
triangle_convoluted_image_d7 = upsampled_image_d7; % the block convolutional filter↙
shold be of the same size
% as of the upsampled image.
for i = 1:size_of_image % for loop condition
    triangle_convoluted_image_d7(i,:) = conv(triangle_convoluted_image_d7(i,:),↙
triangle_filter_d7,'same');
    % Here, the
    % convolutions of the rows of the matrix. convolution operation for rows.
end % for loop ends
for j = 1:size_of_image % for loop condition
    triangle_convoluted_image_d7(:,j) = conv(triangle_convoluted_image_d7(:,j),↙
triangle_filter_d7.', 'same');
    % Here, the
    % convolutions of the rows of the matrix. convolution operation for columns.
end % for loop ends



% Figure plotting for D=7
figure; % figure creates figure graphics objects. figure objects are the individual↙
windows on the screen
% in which MATLAB displays graphical output.
subplot(2,2,1); % subplot(m, n, p) divides the current figure into an m-by-n grid and↙
creates axes in the
% position specified by p. Here, m=n=2, p=1.
imshow(input_image); % It will display the gray-scale image in the figure.
title('Original image'); % It will add the specified title for the current plot.
subplot(2,2,2); % subplot(m, n, p) divides the current figure into an m-by-n grid and↙
creates axes in the
% position specified by p. Here, m=n=p=2.
imshow(uint8(upsampled_image_d7)); % It will display the gray-scale image in the figure↙
and
% it will convert each and every pixel value of the input image into the range of 0 to↙
255.
title('Upsampled image (D=7)'); % It will add the specified title for the current plot.
```

```
subplot(2,2,3); % subplot(m, n, p) divides the current figure into an m-by-n grid and↙
creates axes in the
% position specified by p. Here, m=n=2, p=3.
imshow(uint8(block_convoluted_image_d7)); % It will display the gray-scale image in the↙
figure and
% it will convert each and every pixel value of the input image into the range of 0 to↙
255.
title('Interpolated image using Nearest Neighbour (D=7)'); % It will add the specified↙
title for the current plot.
subplot(2,2,4); % subplot(m, n, p) divides the current figure into an m-by-n grid and↙
creates axes in the
% position specified by p. Here, m=n=2, p=4.
imshow(uint8(triangle_convoluted_image_d7)); % It will display the gray-scale image in↙
the figure and
% it will convert each and every pixel value of the input image into the range of 0 to↙
255.
title('Image using Bilinear interpolation (D=7)'); % It will add the specified title↙
for the current plot.
```