```
%Homework 2 - Q2
% Read the input image (grayscale)
im = imread('tint1.jpg');
im = rgb2gray(im); % Convert to grayscale if the image is in RGB

% Call the dithering functions
jjn_result = jjn(im);  % Jarvis-Judice-Ninke Dithering
fs_result = fs(im, 2);  % Floyd-Steinberg Dithering with 2 levels
```

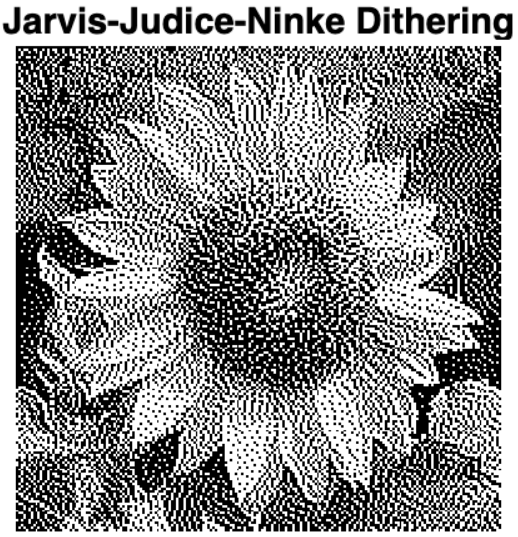
```
% Jarvis-Judice-Ninke Dithering Algorithm
function out = jjn(im)
    height = size(im,1);
    width = size(im,2);
    out = zeros(size(im));
    ed = [0 0 0 7 5; 3 5 7 5 3; 1 3 5 3 1]/48;  % Error diffusion matrix
    z = zeros(size(im) + 4);  % Add padding for boundary handling
    z(3:height+2, 3:width+2) = double(im);  % Place the original image into
padded matrix

    for i = 3:height+2
        for j = 3:width+2
            quant = 255 * (z(i,j) >= 128);  % Quantization: set to 255 or 0
based on threshold 128
            out(i-2, j-2) = quant;  % Store the result in the output matrix
            e = z(i,j) - quant;  % Compute the quantization error
            z(i:i+2, j-2:j+2) = z(i:i+2, j-2:j+2) + e * ed;  % Distribute
the error using the error diffusion matrix
        end
    end
    out = im2uint8(out);  % Convert output to 8-bit unsigned integer
end
```

```
% Floyd-Steinberg Dithering Algorithm
function out = fs(im, k)
    [rs, cs] = size(im);
    ed = [0 0 7; 3 5 1] / 16.0;  % Error diffusion matrix (2x3 matrix)
    z = zeros(rs + 2, cs + 2);  % Add padding for boundary handling
    z(2:rs+1, 2:cs+1) = double(im);  % Place the original image into padded
matrix

    for i = 2:rs+1
        for j = 2:cs+1
            old = z(i,j);
```

```
            new = floor(old / (255 / k)) * (255 / (k - 1));  % Corrected
integer division for MATLAB
            z(i,j) = new;
            E = old - new;  % Compute the quantization error

            % Check boundary to avoid out-of-bound error
            if i+1 <= rs+1 && j+1 <= cs+1
                z(i:i+1, j-1:j+1) = z(i:i+1, j-1:j+1) + E * ed;  %
Distribute the error using the error diffusion matrix
            elseif j+1 <= cs+1
                z(i, j-1:j+1) = z(i, j-1:j+1) + E * ed(1,:);  % Only apply
to existing part
            elseif i+1 <= rs+1
                z(i:i+1, j-1) = z(i:i+1, j-1) + E * ed(:,1);  % Only apply
to existing part
            end
        end
    end
    out = uint8(z(2:rs+1, 2:cs+1));  % Return the result without padding
end
```

```
% Display the original image
figure;
imshow(im);
title('Original Image');
```


```
% Display the Jarvis-Judice-Ninke Dithering result
figure;
imshow(jjn_result);
title('Jarvis-Judice-Ninke Dithering');
```




```
% Display the Floyd-Steinberg Dithering result
figure;
imshow(fs_result);
title('Floyd-Steinberg Dithering');
```