The University of Windsor

ELEC4490: Sensors and Vision Systems

Summer 2020

Project # 1

Friday, June 14, 2020

Emmanuel Mati

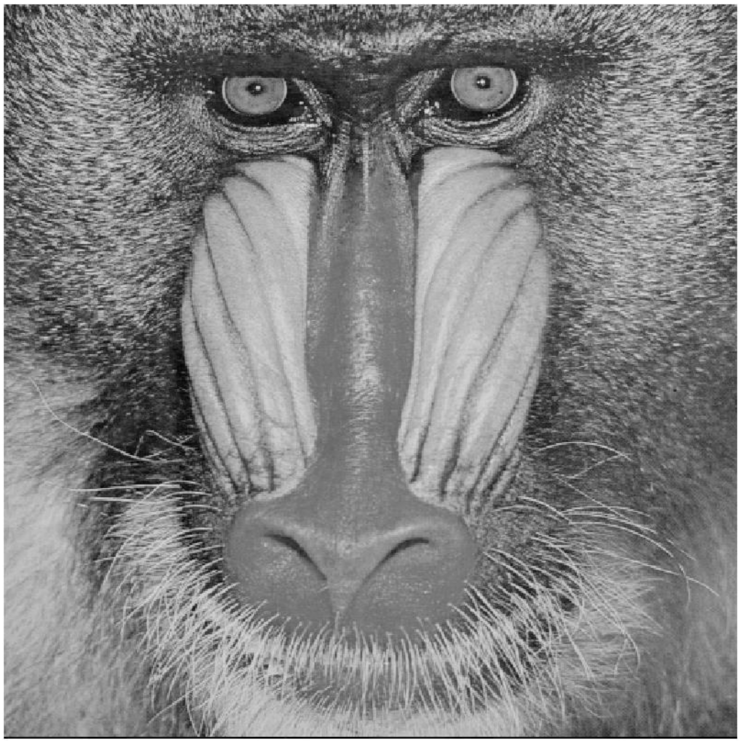
104418019

1a)

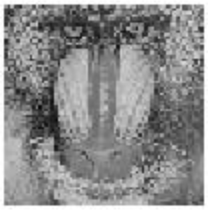
CODE:

```
%Image Processing Assignment
%ELEC-4490 S2020
%Assignment 2
%Author: Emmanuel Mati

%Part 1

clear all; %Deletes all variables
close all; %Closes all figure windows
clc; %Clears the command window

%Retreived image
ImageInput = imread('monkey.gif');



%Recording how many rows and columns the image has
[rows, cols] = size(ImageInput)
%Here we choose how much we want to rezize our image by
newImage= zeros(round(rows/7),round(cols/7));

%Row and column index positions
rind = 1
cind = 1

%Now we shrink the image with two for loops
for row = 1:7:rows
    for col = 1:7:cols
        newImage(rind,cind) = ImageInput(row, col);
        cind = cind + 1;
    end
    cind = 1;
    rind = rind + 1;
end

figure, imshow(ImageInput)
figure, imshow(newImage/255)
```

1b)

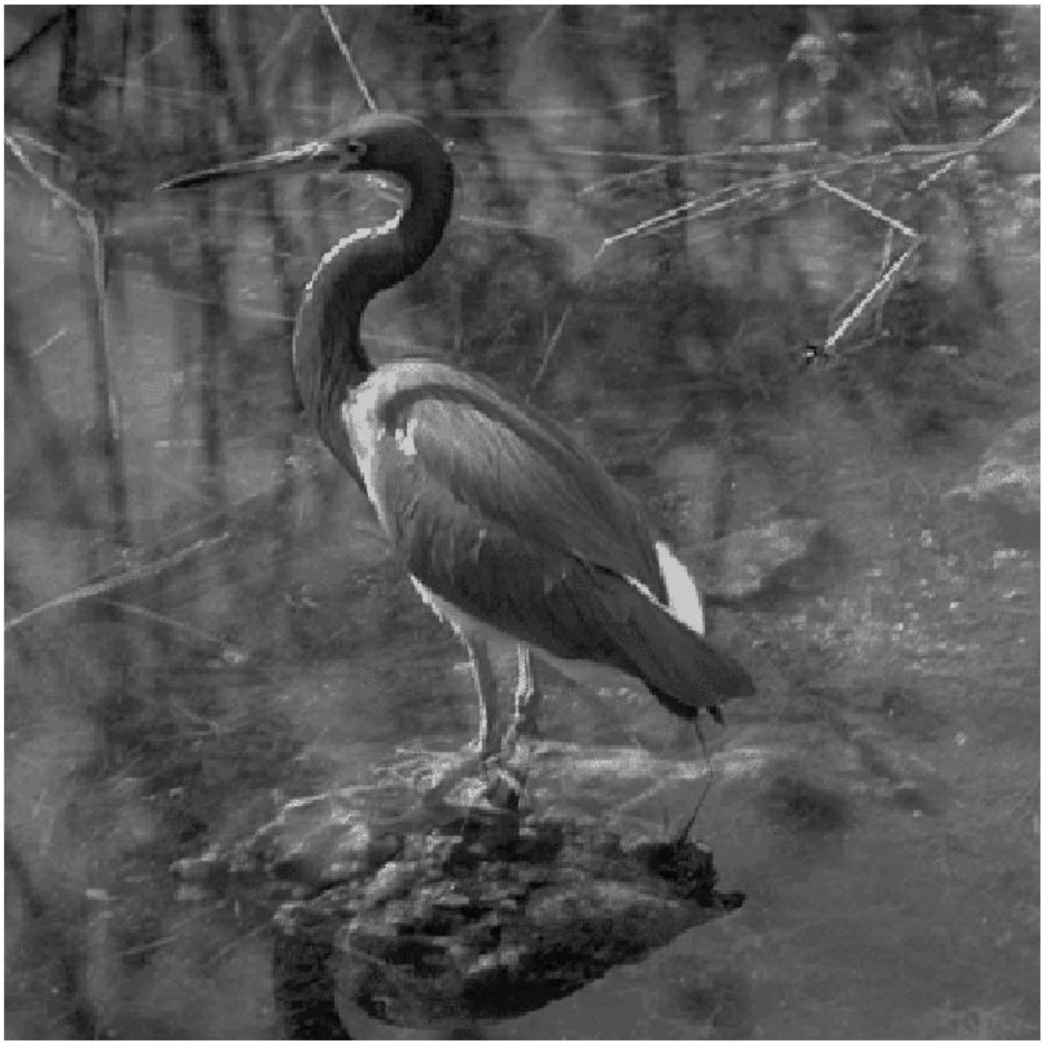


Original 512x512

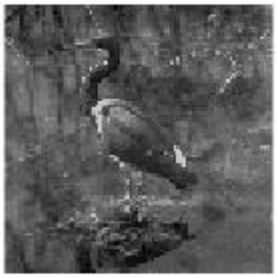


Starts becoming unrecognizable at 64x64

(8 times smaller)

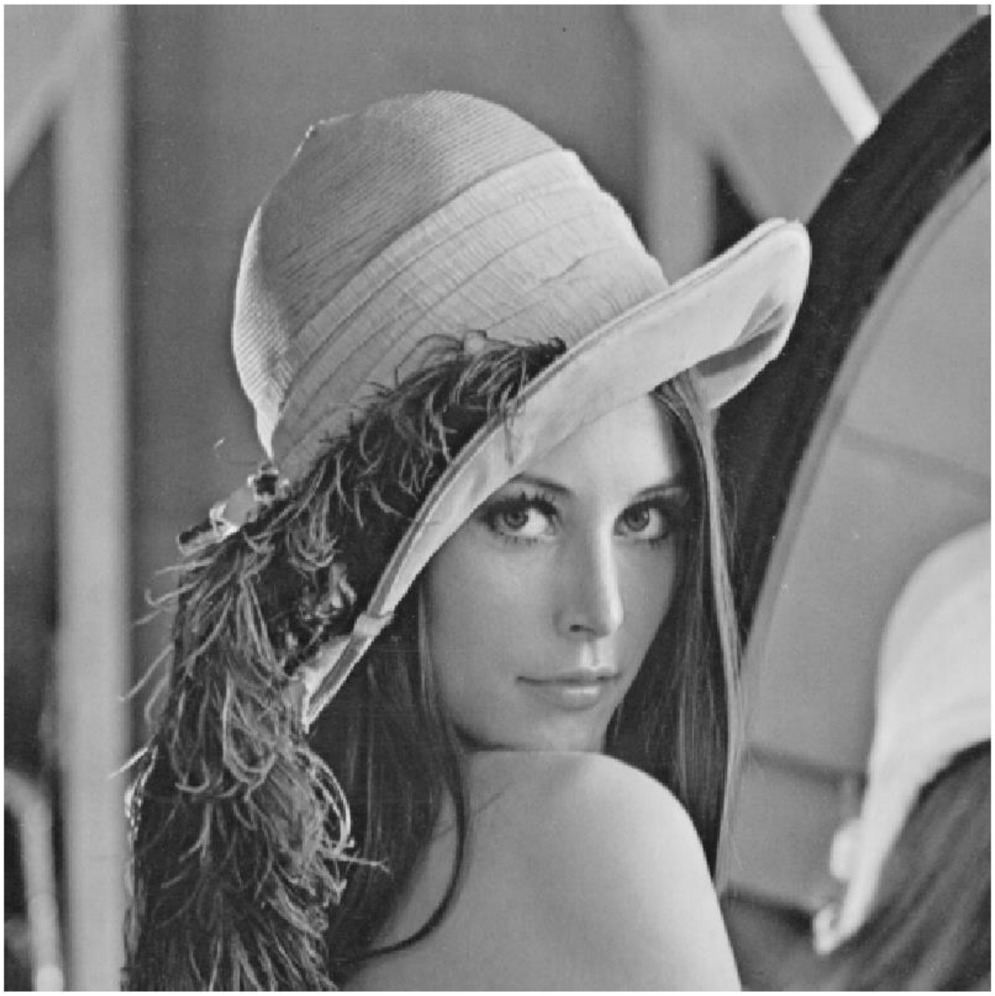


Original 512x512

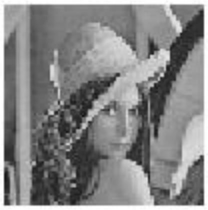


Starts becoming unrecognizable at 86x86

(6 times smaller)



Original 512x512



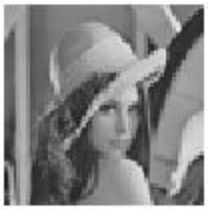
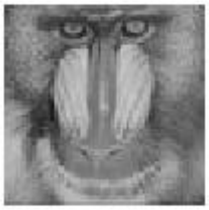
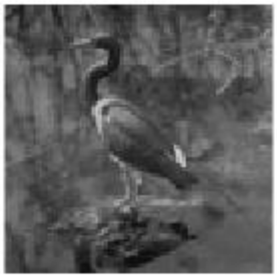
Starts becoming unrecognizable at 64x64

(8 times smaller)

1C)

CODE:

```
%%using imresize
figure, imshow(imresize(imread('lena.png'),[64 64]))
figure, imshow(imresize(imread('monkey.gif'),[64 64]))
figure, imshow(imresize(imread('bird.gif'),[86 86]))
```

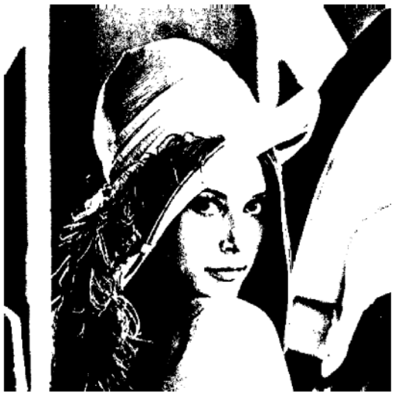
The results with imresize at the same dimensions look clearer

2a)

Code:

```
%Image Processing Assignment
%ELEC-4490 S2020
%Assignment 2
%Author: Emmanuel Mati

% Part 2
clear all; %Deletes all variables
close all; %Closes all figure windows
clc; %Clears the command window

%Retreived image location
GreyImage = imread('lena.png');
imshow(GreyImage);

%Takes image from unit8 array to double array
ImageToBeProcessed = double(GreyImage);

%Taking Resolution of the image
[ImageRow ImageColumn] = size(GreyImage);

% What we want to display in the user prompt
prompt={'Pick a threshold between 0 and 255'};
name='Black and White from Grey';
numlines=1;
defaultanswer={'128'};

%Taking user input and converting it to double
Thresh=str2double(inputdlg(prompt,name,numlines,defaultanswer));

%Blank Image
BlackAndWhite=zeros(ImageRow,ImageColumn);

%loop array from left to right and bottom to top
    for x = 1:ImageRow
        for y = 1:ImageColumn
            %colour selecting
            if ImageToBeProcessed(x,y) < Thresh
                BlackAndWhite(x,y) = 0;
            else
                BlackAndWhite(x,y) = 255;
            end

        end
    end
  imshow(BlackAndWhite)
```

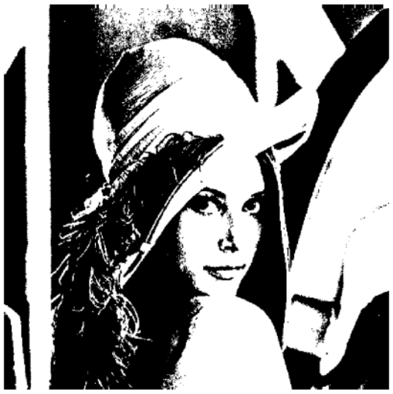
Original Grey Image



Black And White Image at 128(0.5) Threhold

B)

```
%% BW comparison
clear all; %Deletes all variables
close all; %Closes all figure windows
clc; %Clears the command window
GreyImage = imread('lena.png');
BW  = im2bw(GreyImage,0.5);
imshow(BW)
```

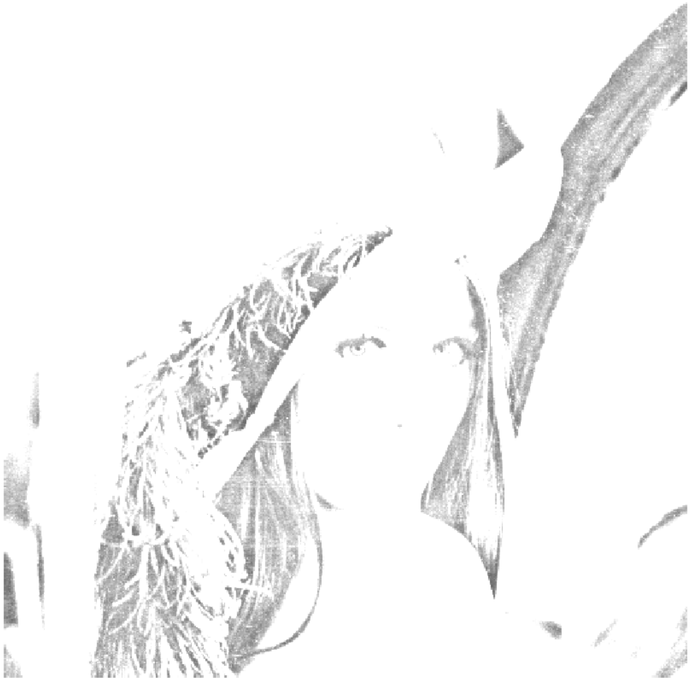


Im2bw result

3)

```
%Image Processing Assignment
%ELEC-4490 S2020
%Assignment 2
%Author: Emmanuel Mati

% Part 3
clear all; %Deletes all variables
close all; %Closes all figure windows
clc; %Clears the command window

%Retreived image location
RetrievedImage = imread('lena.png');
[rows cols] = size(RetrievedImage);

doubleRetrievedImage = double(RetrievedImage);
newImage = zeros(rows, cols);

% What we want to display in the user prompt
prompt={'Enter one of the following intensity levels: 2, 4, 8, 16, 32, 64,
128, 256'};
name='Intensity picker';
numlines=1;
defaultanswer={'256'};

%Taking user input and converting it to double
Thresh=str2double(inputdlg(prompt,name,numlines,defaultanswer));


cind = 1;
rind = 1;

%loops to each pixel and reduces the intensity accordingly
for row = 1:1:rows
    for col = 1:1:cols
        newImage(row, col) = doubleRetrievedImage(row, col)/Thresh;
    end
end

figure, imshow(RetrievedImage)
figure, imshow(newImage)
```

Original



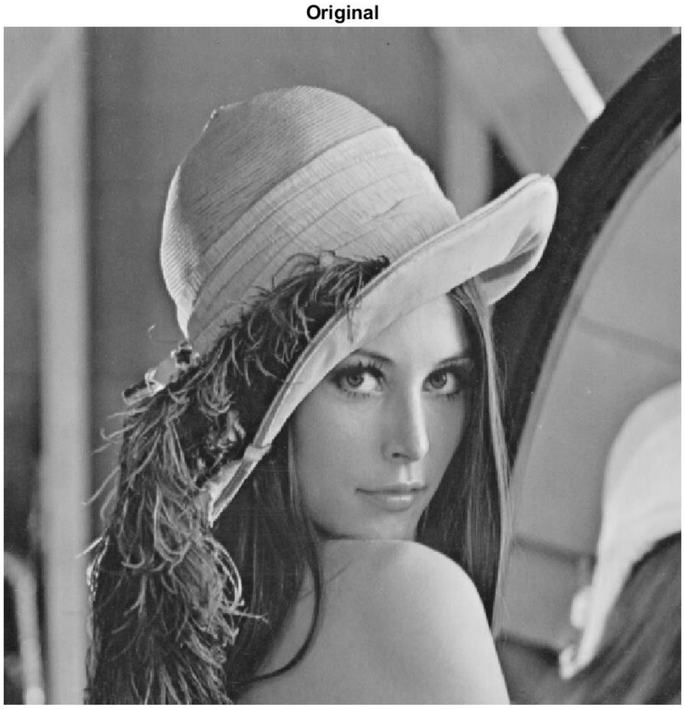
Intensity reduced to 64

4)

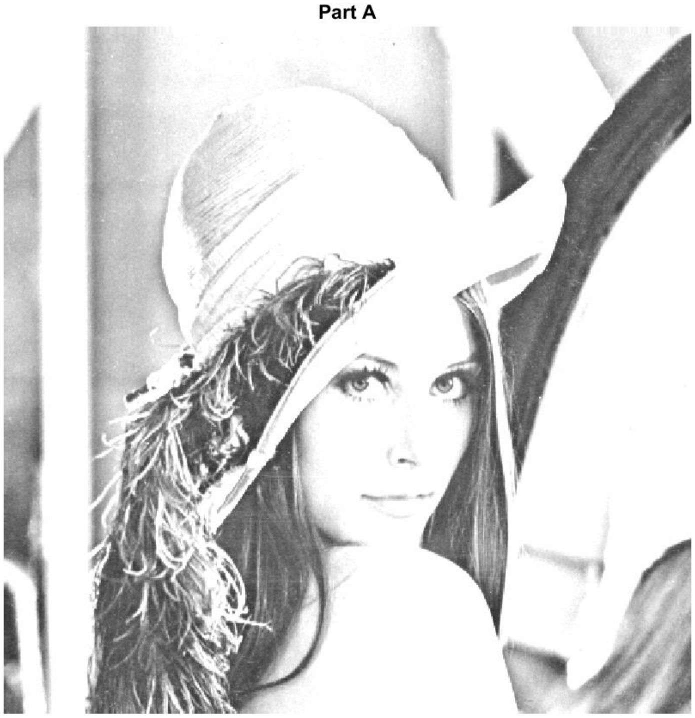
CODE:

```
%%Image Processing Assignment
%ELEC-4490 S2020
%Assignment 2
%Author: Emmanuel Mati

%% Part 4:A, B, C, D
clear all; %Deletes all variables
close all; %Closes all figure windows
clc; %Clears the command window

%Original Image
figure, imshow(imread('lena.png'));
title('Original')

%Retreived 8-bit monochrome image location
Mono8bit = imread('lena.png');
%Double monochrome image
Mono2bit = double(Mono8bit);

%Part A
PartA = Mono8bit + Mono8bit;
figure, imshow(PartA); %Image becomes lighter
title('Part A')

%Part B
PartB = immultiply(Mono8bit, Mono8bit);
figure, imshow(PartA); %Image becomes lighter
title('Part B')

%Part C
PartC =  immultiply(Mono2bit, Mono2bit);
figure, imshow(PartC/255); %Image becomes fully White
title('Part C')

%Part D
PartD =  imdivide(Mono2bit, Mono2bit);
figure, imshow(PartD/255); %Image becomes fully Black
title('Part D')
```

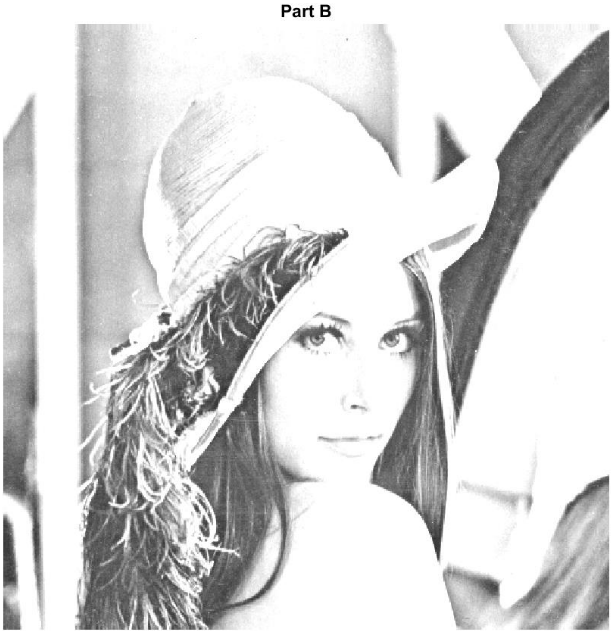
Original



A)Image has become brighter

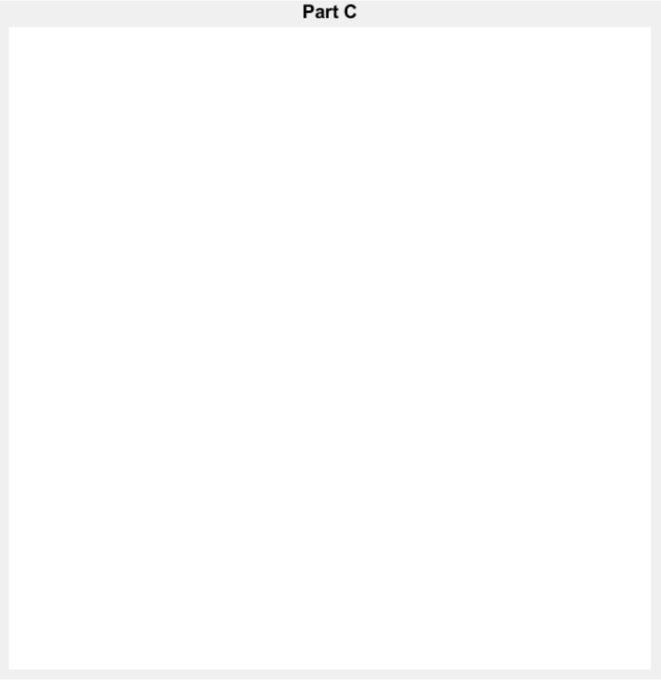


B) Image has the same results as A



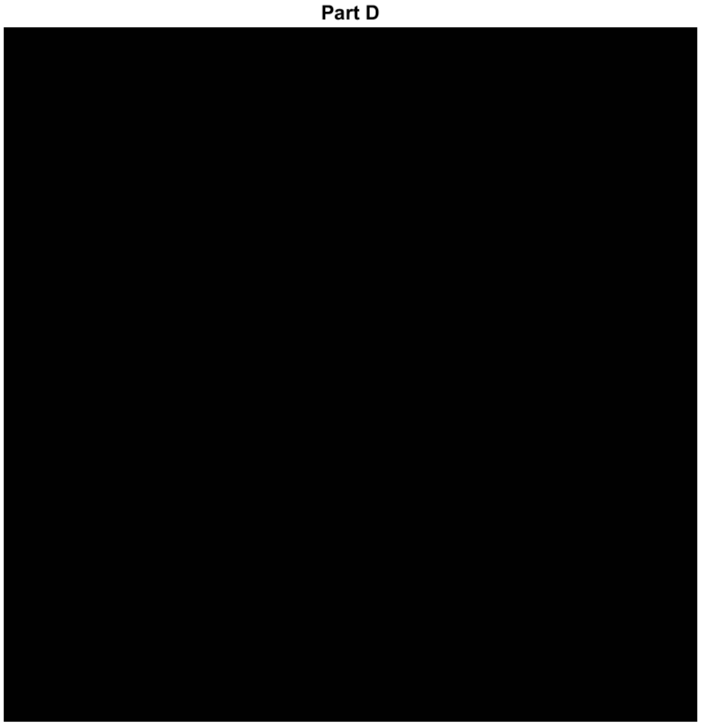
Part B

C) Image has become whited out



Part C

D) Image has become blacked out



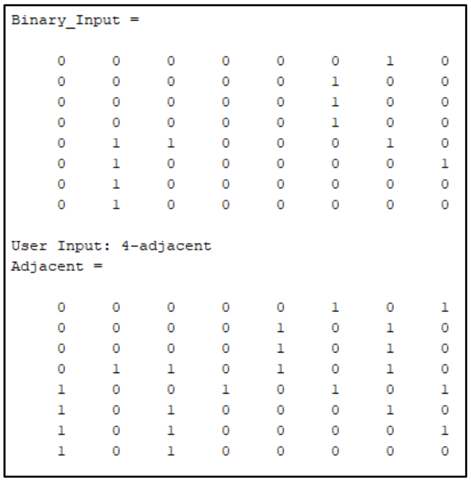
5)

Code:

```
%%Image Processing Assignment
%ELEC-4490 S2020
%Assignment 2
%Author: Emmanuel Mati

%% Part 4: A, B, C
clear all; %Deletes all variables
close all; %Closes all figure windows
clc; %Clears the command window

% What we want to display in the user prompt
prompt={'Enter Desired Adjacency(4, 8, 16, 32, m)'};
name='Adjacenecy picker';
numlines=1;
defaultanswer={'4'};
%Taking user input and converting it to double
UserInput=str2double(inputdlg(prompt,name,numlines,defaultanswer));

%Creating the Binary Array
BinSet =  [ 0 0 0 0 0 0 1 0 ;...
    0 0 0 0 0 1 0 0;...
    0 0 0 0 0 1 0 0;...
    0 0 0 0 0 1 0 0;...
    0 1 1 0 0 0 1 0;...
    0 1 0 0 0 0 0 1;...
    0 1 0 0 0 0 0 0;...
    0 1 0 0 0 0 0 0];
%Getting its size
[rows, columns] = size(BinSet);


%Showing Input
Binary_Input = BinSet
fprintf('User Input: %d-adjacent', UserInput);

%Creating the mask
OurMask = true(rows, columns);
for x=1:1:rows
    for y=1:1:columns
        if BinSet(x,y) == 1
            OurMask(x,y) = false;
        end
    end
end

%Creating our Adjacent Array
Adjacent = zeros(rows, columns);
```

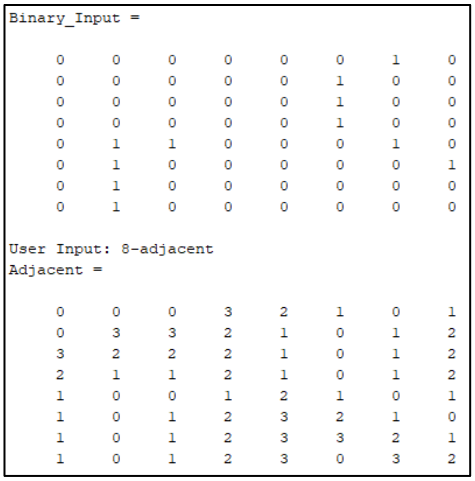
```
%Generate 4-Adjecent
for x=1:1:rows
    for y=1:1:columns
        if OurMask(x,y) == false
            Adjacent(x,y)= 0;
        elseif OurMask(x,y) == true
            %checks up
            if x ~= 1 & OurMask(x-1,y)== false
               Adjacent(x,y)= 1;
            %checks up
            elseif  x ~= rows & OurMask(x+1,y)== false
                Adjacent(x,y)= 1;
            %checks left
            elseif  y ~= 1 & OurMask(x,y-1)== false
                Adjacent(x,y)= 1;
            %checks right
            elseif  y ~= columns & OurMask(x,y+1)== false
                Adjacent(x,y)= 1;
            end
        end
    end
end

%Calculates adjacency above 4
if UserInput ~= 4
    for t = 2:1:log2(UserInput)
        for x=1:1:rows
            for y=1:1:columns
                if OurMask(x,y) == true & Adjacent(x,y) == 0 & t ~= 0
                    %checks up
                    if x ~= 1 & Adjacent(x-1,y) == t - 1 & OurMask(x-1,y) ==
true
                       Adjacent(x,y)= t;
                    %checks up
                    elseif  x ~= rows & Adjacent(x+1,y) == t - 1 &
OurMask(x+1,y) == true
                        Adjacent(x,y)= t;
                    %checks left
                    elseif  y ~= 1 & Adjacent(x,y-1) == t - 1 & OurMask(x,y-
1) == true
                        Adjacent(x,y)= t;
                    %checks right
                    elseif  y ~= columns & Adjacent(x,y+1) == t - 1 &
OurMask(x,y+1) == true
                        Adjacent(x,y)= t;
                    end
                end
            end
        end
    end
end

Adjacent
```

A) 4-Adjacent



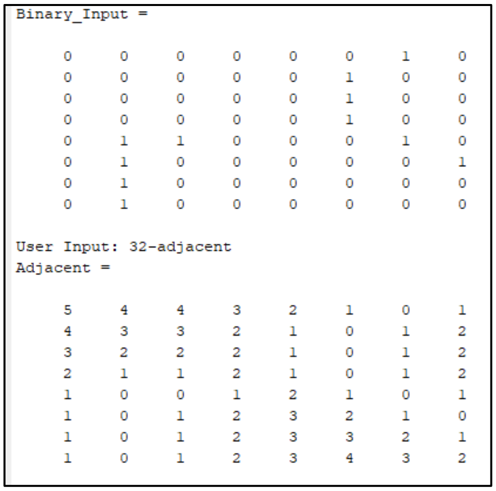
```
Binary_Input =

     0     0     0     0     0     0     1     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     1     0     0
     0     1     1     0     0     0     1     0
     0     1     0     0     0     0     0     1
     0     1     0     0     0     0     0     0
     0     1     0     0     0     0     0     0

User Input: 4-adjacent
Adjacent =

     0     0     0     0     0     1     0     1
     0     0     0     0     1     0     1     0
     0     0     0     0     1     0     1     0
     0     1     1     0     1     0     1     0
     1     0     0     1     0     1     0     1
     1     0     1     0     0     0     1     0
     1     0     1     0     0     0     0     1
     1     0     1     0     0     0     0     0
```

B) 8-Adjacent



```
Binary_Input =

     0     0     0     0     0     0     1     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     1     0     0
     0     1     1     0     0     0     1     0
     0     1     0     0     0     0     0     1
     0     1     0     0     0     0     0     0
     0     1     0     0     0     0     0     0

User Input: 8-adjacent
Adjacent =

     0     0     0     3     2     1     0     1
     0     3     3     2     1     0     1     2
     3     2     2     2     1     0     1     2
     2     1     1     2     1     0     1     2
     1     0     0     1     2     1     0     1
     1     0     1     2     3     2     1     0
     1     0     1     2     3     3     2     1
     1     0     1     2     3     0     3     2
```

C) m-Adjacent(32)



```
Binary_Input =

     0     0     0     0     0     0     1     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     1     0     0
     0     1     1     0     0     0     1     0
     0     1     0     0     0     0     0     1
     0     1     0     0     0     0     0     0
     0     1     0     0     0     0     0     0

User Input: 32-adjacent
Adjacent =

     5     4     4     3     2     1     0     1
     4     3     3     2     1     0     1     2
     3     2     2     2     1     0     1     2
     2     1     1     2     1     0     1     2
     1     0     0     1     2     1     0     1
     1     0     1     2     3     2     1     0
     1     0     1     2     3     3     2     1
     1     0     1     2     3     4     3     2
```