

**Image Processing
(01CE0507)**

**Department of Computer Engineering
5th Semester**

**Lab Manual
(July-Dec 2022)**

Index

Lab	Programs	Date	Signature
1	Study of matlab image processing toolkit and various commands on matlab.		
2	Point processing in spatial domain <ul style="list-style-type: none"> a. Negation of an image b. Thresholding of an image c. Contrast Stretching of an image 		
3	Write a program for histogram equalization.		
4	Write a program to apply various filtering techniques in matlab. <ul style="list-style-type: none"> a. Low pass filtering b. High pass filtering c. Median filtering 		
5	Write a program for image segmentation <ul style="list-style-type: none"> a. Local thresholding b. Global thresholding 		
6	Write a program for color image processing <ul style="list-style-type: none"> a. Color approximation b. Quantization 		
7	Write a program, for Image restoration <ul style="list-style-type: none"> a. Facial Images b. Texture Images 		
8	Write a program for edge detection.		
9	Write a program for smoothening and sharpening for 8-bit color image.		
10	Write a program to implement morphological operations.		

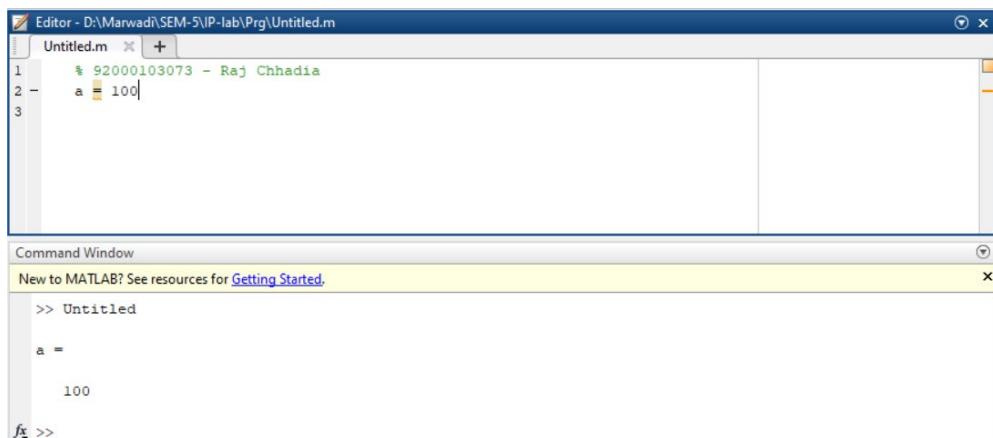
Practical 1

Aim: Study of matlab image processing toolkit and various commands on matlab.

- **Introduction to Various component of MATLAB Tool like editor, command window, workspace.**
- **Basic Syntax, Variables, Commands**
 - **Use of Semicolon (;) in MATLAB**

1) Without Semicolon

- Use semicolons to separate rows in an array creation command, or to suppress the output display of a line of code.



The screenshot shows the MATLAB environment. The Editor window displays the following code:

```

Editor - D:\Marwadi\SEM-5\P-lab\Prg\Untitled.m
Untitled.m + 
1 % 92000103073 - Raj Chhadia
2 a = 100
3

```

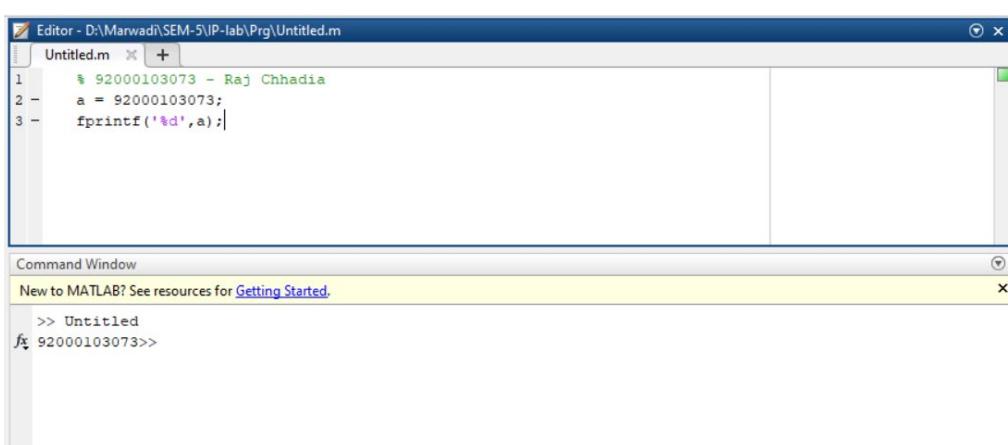
The Command Window shows the results of running the script:

```

Command Window
New to MATLAB? See resources for Getting Started.
>> Untitled
a =
100
fz >>

```

2) With Semicolon



The screenshot shows the MATLAB environment. The Editor window displays the following code:

```

Editor - D:\Marwadi\SEM-5\P-lab\Prg\Untitled.m
Untitled.m + 
1 % 92000103073 - Raj Chhadia
2 a = 92000103073;
3 fprintf('%d',a);

```

The Command Window shows the results of running the script:

```

Command Window
New to MATLAB? See resources for Getting Started.
>> Untitled
fz 92000103073>>

```

- **Adding Comments**

→ Comments are non-executable text within the body of a program. They can be declared by ‘%’



```

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\Untitled.m
Untitled.m x +
1 % 92000103073 - Raj Chhadia
2 % this is a comment
3

```

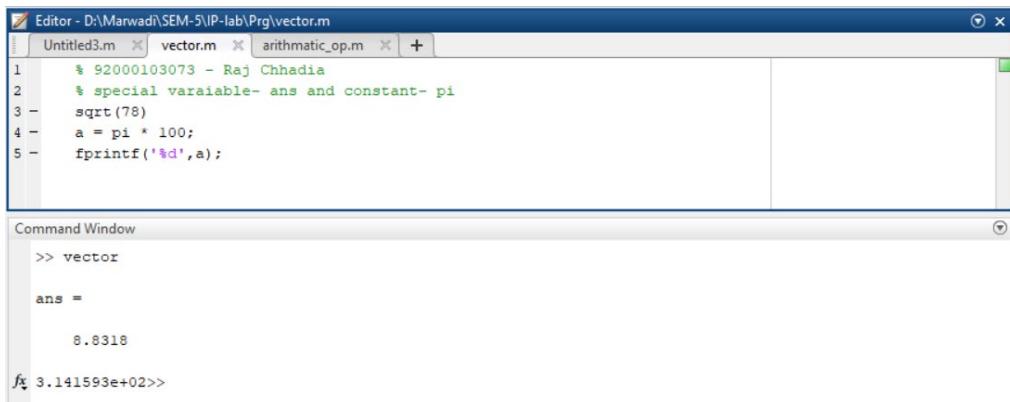
- **Commonly used Operators and Special Characters**

Operator	Purpose
+	Plus; addition operator.
-	Minus; subtraction operator.
*	Scalar and matrix multiplication operator.
.*	Array multiplication operator.
^	Scalar and matrix exponentiation operator.
.^	Array exponentiation operator.
\	Left-division operator.
/	Right-division operator.
.\	Array left-division operator.
./	Array right-division operator.
:	Colon; generates regularly spaced elements and represents an entire row or column.
()	Parentheses; encloses function arguments and array indices; overrides precedence.
[]	Brackets; enclosures array elements.
.	Decimal point.
...	Ellipsis; line-continuation operator
,	Comma; separates statements and elements in a row
;	Semicolon; separates columns and suppresses display.
%	Percent sign; designates a comment and specifies formatting.
'	Quote sign and transpose operator.
.	Nonconjugated transpose operator.
=	Assignment operator.

Name	Meaning
ans	Most recent answer.
eps	Accuracy of floating-point precision.
i,j	The imaginary unit $\sqrt{-1}$.
Inf	Infinity.
NaN	Undefined numerical result (not a number).
pi	The number π

○ Special Variables and Constants

→ sqrt, pi are some of the special constants and ans is one of the special variables to store the answer.



The screenshot shows the MATLAB environment. In the Editor window, a script named 'vector.m' contains the following code:

```

1 % 92000103073 - Raj Chhadia
2 % special variable- ans and constant- pi
3 -
4 - sqrt(78)
5 - a = pi * 100;
6 - fprintf('%d',a);

```

In the Command Window, the command 'vector' is run, resulting in:

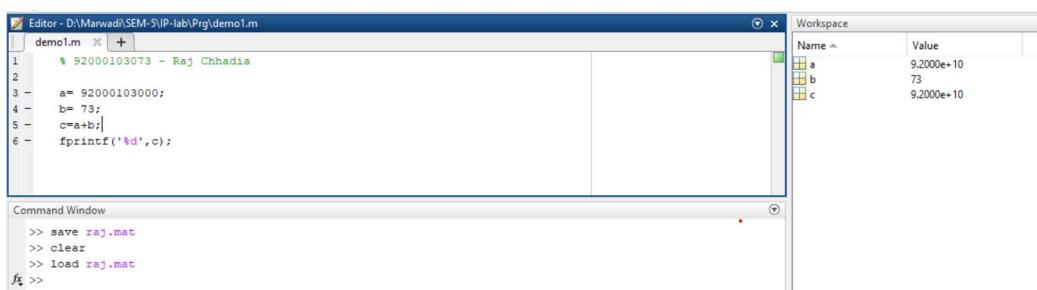
```

>> vector
ans =
8.8318
f 3.141593e+02>>

```

○ save and load Command

→ save command saves workspace variables in a file. load command loads workspace variables from a file.



The screenshot shows the MATLAB environment. In the Editor window, a script named 'demo1.m' contains the following code:

```

1 % 92000103073 - Raj Chhadia
2 -
3 - a= 92000103000;
4 - b= 73;
5 - c=a+b;
6 - fprintf('%d',c);

```

In the Command Window, the commands 'save raj.mat', 'clear', and 'load raj.mat' are run, followed by a prompt for input:

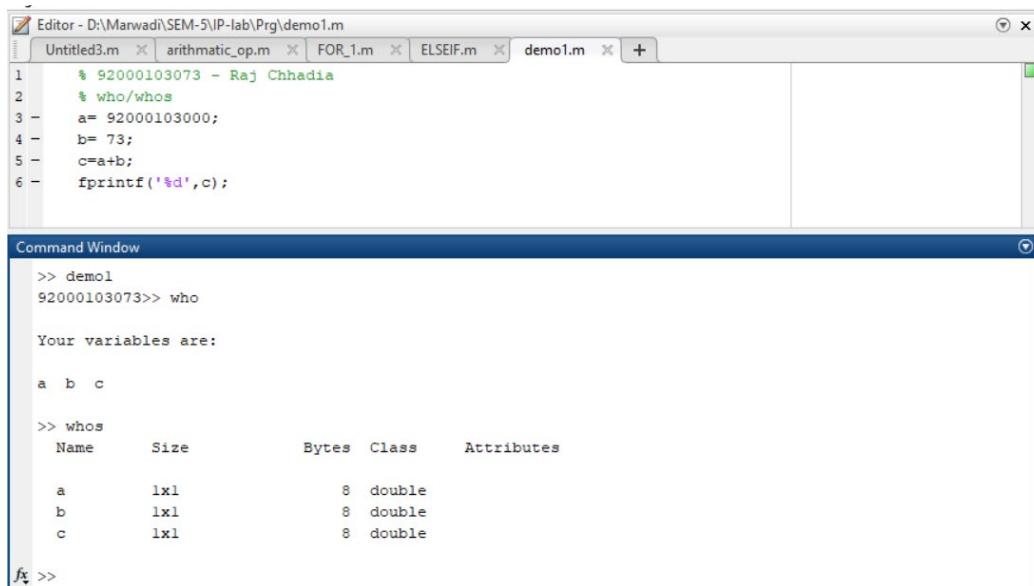
```

>> save raj.mat
>> clear
>> load raj.mat
f>>

```

To the right, the Workspace browser shows the variables 'a', 'b', and 'c' with their respective values.

- **who and whos command**



The screenshot shows the MATLAB environment. The Editor window displays a script named demo1.m with the following code:

```

1 % 92000103073 - Raj Chhadia
2 % who/whos
3 a= 92000103000;
4 b= 73;
5 c=a+b;
6 fprintf('%d',c);

```

The Command Window shows the execution of the script and the output of the who and whos commands:

```

>> demo1
92000103073>> who

Your variables are:

a b c

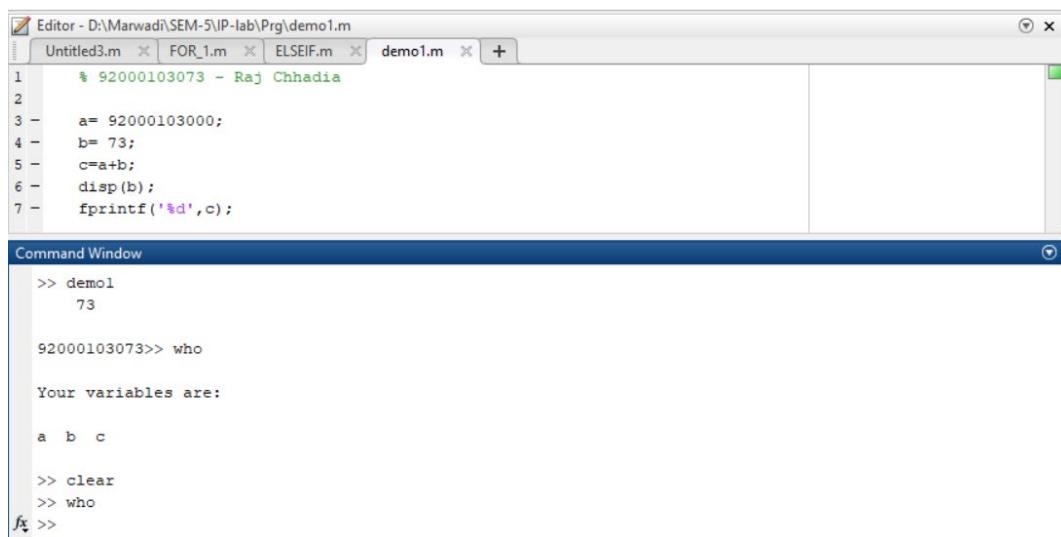
>> whos
  Name      Size            Bytes  Class       Attributes
  a            1x1                 8  double
  b            1x1                 8  double
  c            1x1                 8  double

>>

```

- **clear and clc Command**

→ clc clears command window. clear removes variables from memory.



The screenshot shows the MATLAB environment. The Editor window displays a script named demo1.m with the following code:

```

1 % 92000103073 - Raj Chhadia
2
3 a= 92000103000;
4 b= 73;
5 c=a+b;
6 disp(b);
7 fprintf('%d',c);

```

The Command Window shows the execution of the script and the use of clear and clc commands:

```

>> demo1
    73

92000103073>> who

Your variables are:

a b c

>> clear
>> who
>>

```

- **disp and fprintf command**

→ disp displays contents of an array / string. fprintf performs formatted writes to screen.



The screenshot shows the MATLAB environment. The Editor window contains the following code:

```

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\demo1.m
Untitled3.m x FOR_1.m x ELSEIF.m x demo1.m x +
1 % 92000103073 - Raj Chhadia
2 % disp/fprintf
3 a= 92000103000;
4 b= 73;
5 c=a+b;
6 disp(b);
7 fprintf('%d',c);

```

The Command Window shows the output of running the script:

```

Command Window
>> demo1
73

fx 92000103073>>

```

- **input command**



The screenshot shows the MATLAB environment. The Editor window contains the following code:

```

demo1.m x demo.txt x +
1 % 92000103073 - Raj Chhadia
2 b= 73;
3 disp(b);
4 fileID = fopen('demo.txt','r');
5 formatSpec = '%s';
6 e = fscanf(fileID,formatSpec)
7 fprintf('%d',b);

```

The Command Window shows the output of running the script:

```

Command Window
>> demo1
73

e =
RajChhadia

fx 73>> |

```

- **M-Files**

→ Script files are program files with .m extension. In these files, you write series of commands, which you want to execute together.



```

demo1.m  demo.txt  +
1 % 92000103073 - Raj Chhadia
2 a = 5; b = 7;
3 c = a + b
4 d = c + sin(b)
5 e = 5 * d

```

Command Window

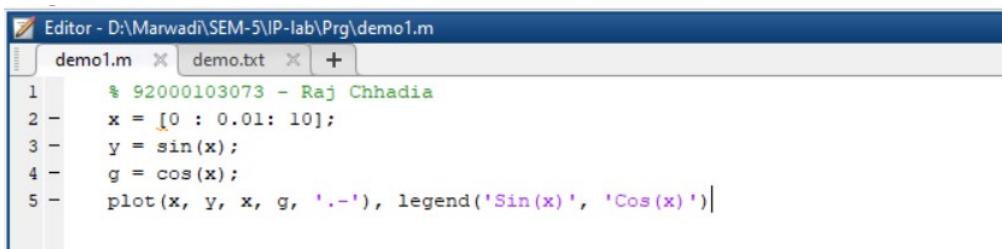
```

>> demol
c =
    12
d =
    12.6570
e =
    63.2849
fx >>

```

- **Plotting Commands**

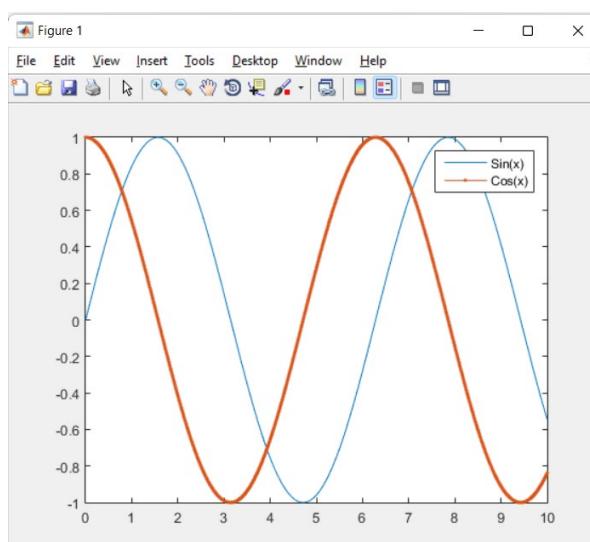
→ plot command is used to plot a graph or curve.



```

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\demo1.m
demo1.m  demo.txt  +
1 % 92000103073 - Raj Chhadia
2 x = [0 : 0.01: 10];
3 y = sin(x);
4 g = cos(x);
5 plot(x, y, x, g, '-.'), legend('Sin(x)', 'Cos(x)')

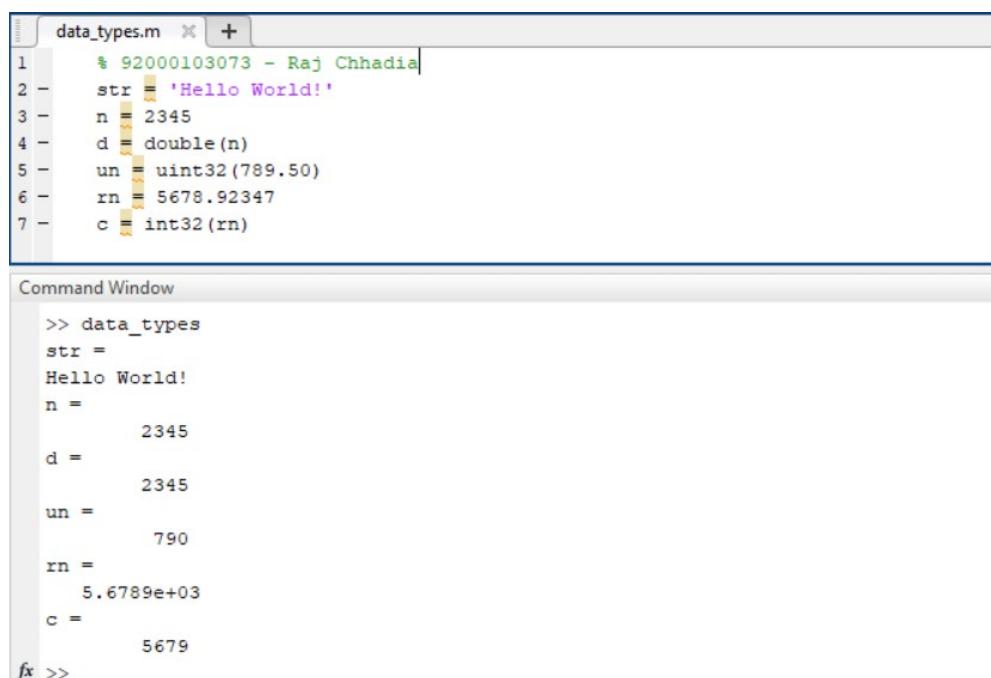
```



➤ Data Types in MATLAB

○ List of Data Types

No.	Data Types
1.	int8
2.	uint8
3.	int32
4.	uint32
5.	int64
6.	uint64
7.	double
8.	char
9.	cell array
10.	structure



The screenshot shows the MATLAB IDE interface. The code editor window contains a script named 'data_types.m' with the following content:

```

1 % 92000103073 - Raj Chhadia
2 str = 'Hello World!'
3 n = 2345
4 d = double(n)
5 un = uint32(789.50)
6 rn = 5678.92347
7 c = int32(rn)

```

The command window below shows the execution of the script and the resulting variable assignments:

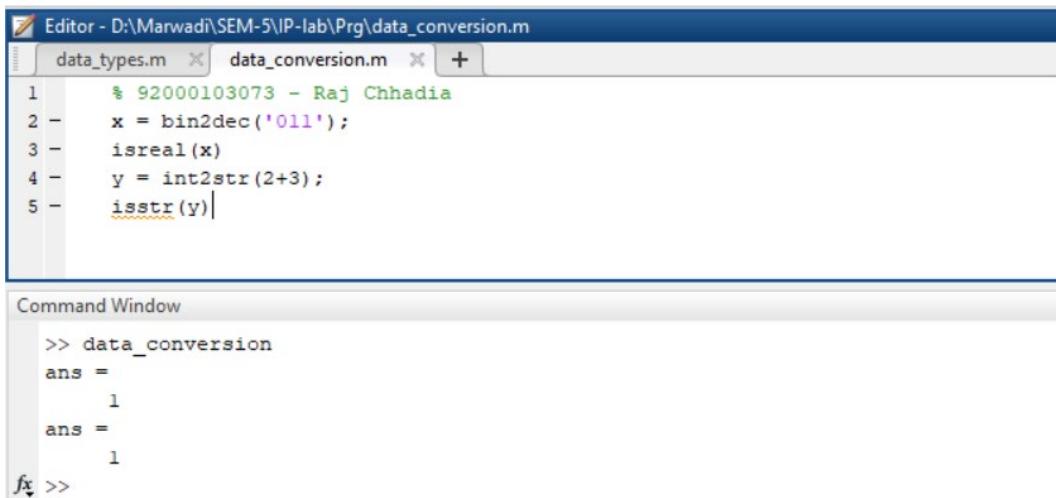
```

>> data_types
str =
Hello World!
n =
2345
d =
2345
un =
790
rn =
5.6789e+03
c =
5679
>>

```

- Data Type Conversion

Function	Purpose
char	Convert to character array (string)
int2str	Convert integer data to string
num2str	Convert number to string
str2double	Convert string to double-precision value
str2num	Convert string to number
bin2dec	Convert binary number to decimal number
dec2base	Convert decimal to base N number in string
dec2bin	Convert decimal to binary number in string
hex2dec	Convert hexadecimal number string to decimal number
mat2cell	Convert array to cell array with potentially different sized cells
cellstr	Create cell array of strings from character array



The screenshot shows the MATLAB environment. The Editor window displays the following code:

```

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\data_conversion.m
data_types.m  data_conversion.m  +
1 % 92000103073 - Raj Chhadia
2 x = bin2dec('011');
3 isreal(x)
4 y = int2str(2+3);
5 isstr(y)

```

The Command Window shows the execution of the script:

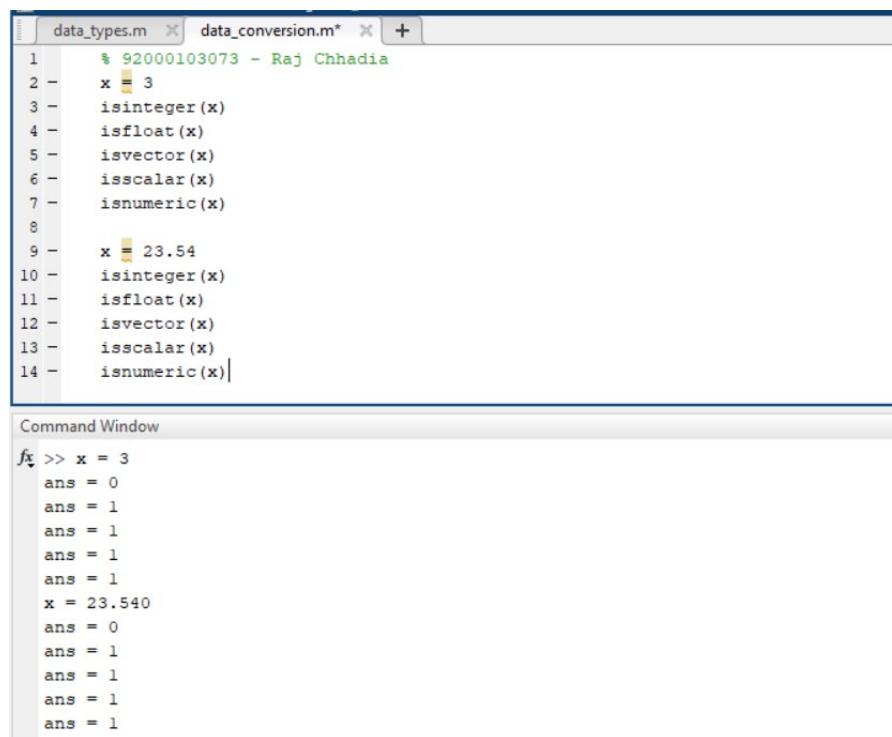
```

Command Window
>> data_conversion
ans =
  1
ans =
  1
fx >>

```

- **Determination of Data Types**

Function	Purpose
isinteger	Determine if input is integer
isfloat	Determine if input is float
isvector	Determine if input is vector
isscalar	Determine if input is scalar
isnumeric	Determine if input is numeric array



The screenshot shows the MATLAB IDE interface. The top window is titled 'data_types.m' and contains the following MATLAB code:

```

1 % 92000103073 - Raj Chhadia
2 x = 3
3 isinteger(x)
4 isfloat(x)
5 isvector(x)
6 isscalar(x)
7 isnumeric(x)
8
9 x = 23.54
10 isinteger(x)
11 isfloat(x)
12 isvector(x)
13 isscalar(x)
14 isnumeric(x)

```

The bottom window is titled 'Command Window' and shows the output of the code execution:

```

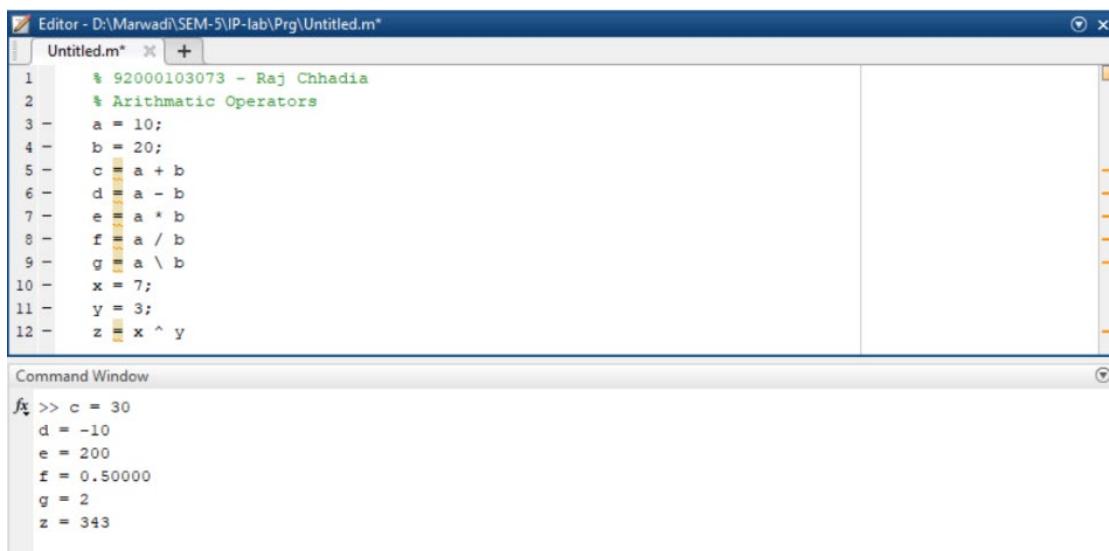
fx >> x = 3
ans = 0
ans = 1
ans = 1
ans = 1
ans = 1
x = 23.540
ans = 0
ans = 1
ans = 1
ans = 1
ans = 1

```

➤ Operators in MATLAB

- Types of Operators
 - Arithmetic Operators

No.	Symbol	Description
1.	+	Addition or unary plus. A+B adds the values stored in variables A and B.
2.	-	Subtraction or unary minus. A-B subtracts value of B from A.
3.	*	Matrix multiplication. C = A*B is the linear algebraic product of the matrices A and B.
4.	.*	Array multiplication. A.*B is the element-by-element product of the arrays A and B.
5.	/	Slash or matrix right division. B/A is roughly the same as B*inv(A).
6.	./	Array right division. A./B is the matrix with elements A(i,j)/B(i,j).
7.	\	Backslash or matrix left division. If A is a square matrix, A\B is roughly the same as inv(A)*B.
8.	.\ .\	Array left division. A.\B is the matrix with elements B(i,j)/A(i,j).
9.	^	Matrix power. X^p is X to the power p, if p is a scalar. If p is an integer, the power is computed by repeated squaring.
10.	.^	Array power. A.^B is the matrix with elements A(i,j) to the B(i,j) power.
11.	'	Matrix transpose. A' is the linear algebraic transpose of A.
12.	.'	Array transpose. A.' is the array transpose of A.



The screenshot shows the MATLAB environment. The Editor window displays a script named Untitled.m* with the following code:

```

1 % 92000103073 - Raj Chhadia
2 % Arithmatic Operators
3 a = 10;
4 b = 20;
5 c = a + b;
6 d = a - b;
7 e = a * b;
8 f = a / b;
9 g = a \ b;
10 x = 7;
11 y = 3;
12 z = x ^ y;

```

The Command Window below shows the results of running the script:

```

>> c = 30
d = -10
e = 200
f = 0.50000
g = 2
z = 343

```

▪ Relational Operators

Symbol	Description
==	Determine Equality
>=	Determine greater than or equal to
>	Determine greater than
<=	Determine less than or equal to
<	Determine less than
~=	Determine inequality
isequal	Determine array equality.
isequaln	Determine array equality, treating NaN values as equal



The screenshot shows the MATLAB environment. The Editor window displays a script named relational_op.m with the following code:

```

1 % 92000103073 - Raj Chhadia
2 % Relational Operators
3 a = 100;
4 b = 200;
5 if (a >= b)
6 max = a
7 else
8 max = b
9 end

```

The Command Window below shows the result of running the script:

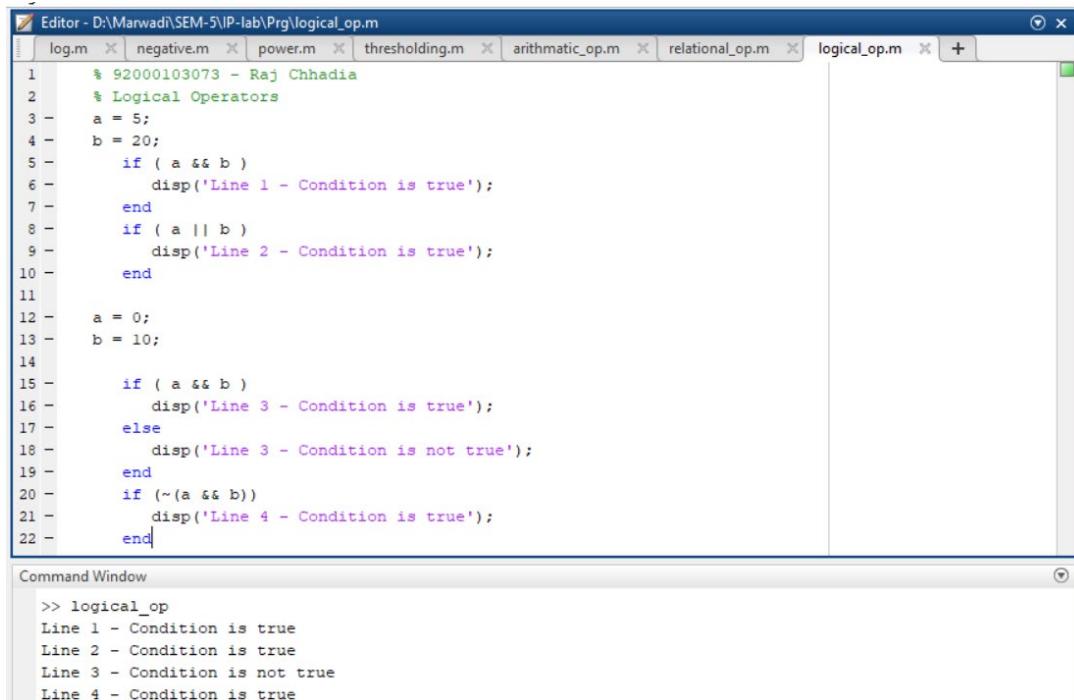
```

>> max = 200

```

■ Logical Operators

Symbol	Description
&& ,	Short Circuit Operations
&	Logical AND
~	Logical NOT
	Logical OR
xor	Logical XOR
all	Determine if all array elements are nonzero or not
any	Determine if any value is zero in array
false	Logical 0 (low)
true	Logical 1(high)



The screenshot shows the MATLAB environment. The Editor window displays a script named 'logical_op.m' with the following code:

```

1 % 92000103073 - Raj Chhadia
2 % Logical Operators
3 a = 5;
4 b = 20;
5 if ( a && b )
6 disp('Line 1 - Condition is true');
7 end
8 if ( a || b )
9 disp('Line 2 - Condition is true');
10 end
11
12 a = 0;
13 b = 10;
14
15 if ( a && b )
16 disp('Line 3 - Condition is true');
17 else
18 disp('Line 3 - Condition is not true');
19 end
20 if (~(a && b))
21 disp('Line 4 - Condition is true');
22 end

```

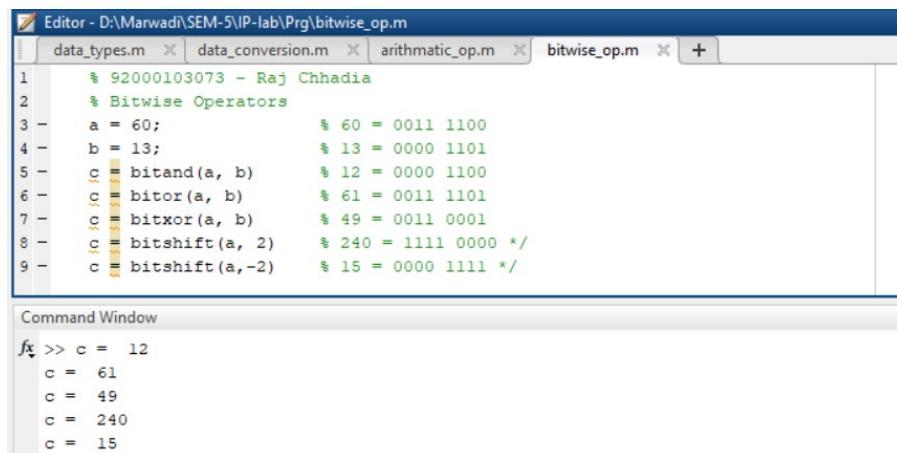
The Command Window below shows the output of running the script:

```

>> logical_op
Line 1 - Condition is true
Line 2 - Condition is true
Line 3 - Condition is not true
Line 4 - Condition is true

```

■ Bitwise Operations



The screenshot shows the MATLAB Editor and Command Window. The Editor displays a script named 'bitwise_op.m' with the following code:

```

1 % 92000103073 - Raj Chhadia
2 % Bitwise Operators
3 a = 60; % 60 = 0011 1100
4 b = 13; % 13 = 0000 1101
5 c = bitand(a, b); % 12 = 0000 1100
6 c = bitor(a, b); % 61 = 0011 1101
7 c = bitxor(a, b); % 49 = 0011 0001
8 c = bitshift(a, 2); % 240 = 1111 0000 */
9 c = bitshift(a,-2); % 15 = 0000 1111 */

```

The Command Window shows the execution of the script:

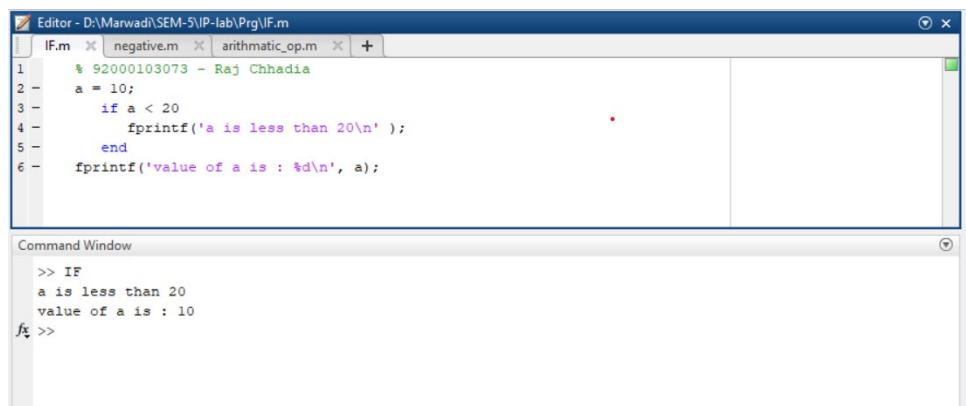
```

>> c =
c = 12
c = 61
c = 49
c = 240
c = 15

```

➤ Decision Making in MATLAB

- if ... end statement



The screenshot shows the MATLAB Editor and Command Window. The Editor displays a script named 'IF.m' with the following code:

```

1 % 92000103073 - Raj Chhadia
2 a = 10;
3 if a < 20
4 fprintf('a is less than 20\n');
5 end
6 fprintf('value of a is : %d\n', a);

```

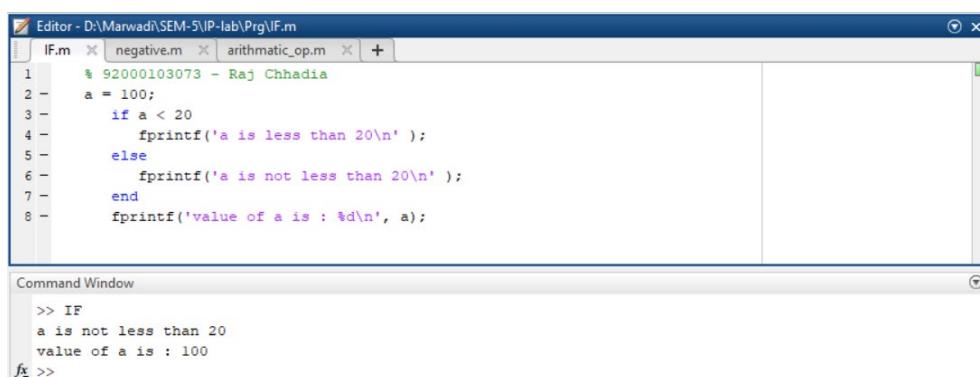
The Command Window shows the execution of the script:

```

>> IF
a is less than 20
value of a is : 10

```

- if...else...end statement



The screenshot shows the MATLAB Editor and Command Window. The Editor displays a script named 'IF.m' with the following code:

```

1 % 92000103073 - Raj Chhadia
2 a = 100;
3 if a < 20
4 fprintf('a is less than 20\n');
5 else
6 fprintf('a is not less than 20\n');
7 end
8 fprintf('value of a is : %d\n', a);

```

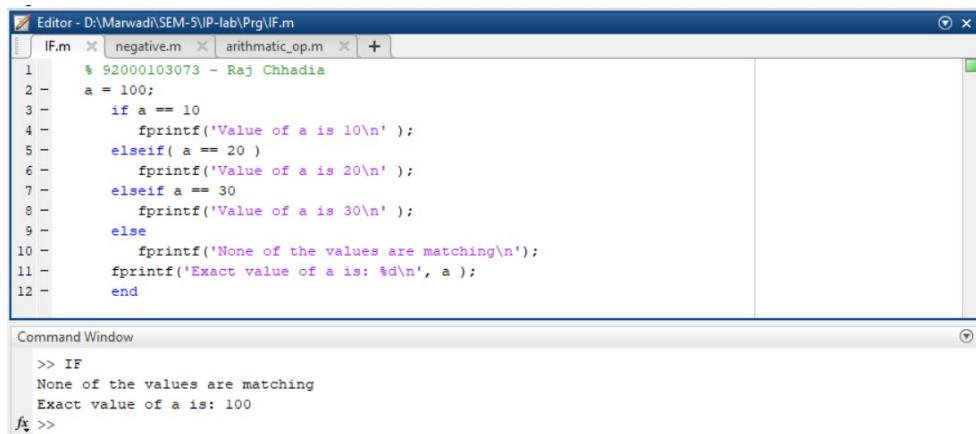
The Command Window shows the execution of the script:

```

>> IF
a is not less than 20
value of a is : 100

```

- If... elseif...elseif...else...end statements



Editor - D:\Marwadi\SEM-5\IP-lab\Prg\IF.m

```

1 % 92000103073 - Raj Chhadia
2 a = 100;
3 if a == 10
4 fprintf('Value of a is 10\n');
5 elseif( a == 20 )
6 fprintf('Value of a is 20\n');
7 elseif a == 30
8 fprintf('Value of a is 30\n');
9 else
10 fprintf('None of the values are matching\n');
11 fprintf('Exact value of a is: %d\n', a);
12 end

```

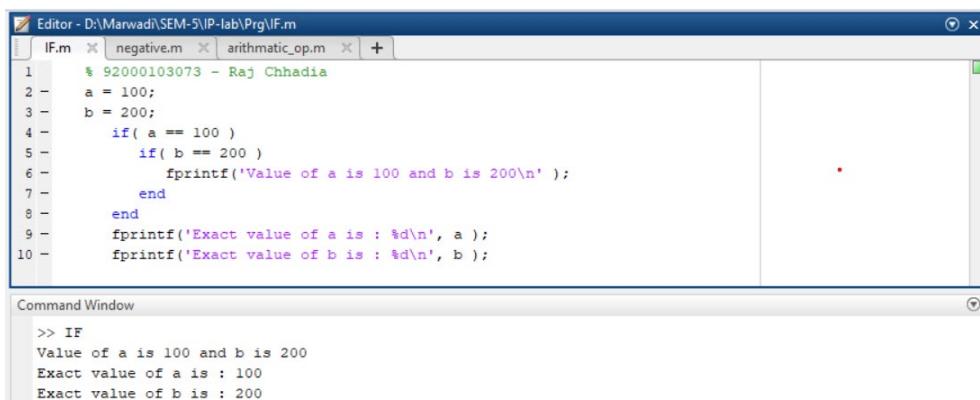
Command Window

```

>> IF
None of the values are matching
Exact value of a is: 100
fx >>

```

- nested if statements



Editor - D:\Marwadi\SEM-5\IP-lab\Prg\IF.m

```

1 % 92000103073 - Raj Chhadia
2 a = 100;
3 b = 200;
4 if( a == 100 )
5 if( b == 200 )
6 fprintf('Value of a is 100 and b is 200\n');
7 end
8 end
9 fprintf('Exact value of a is : %d\n', a );
10 fprintf('Exact value of b is : %d\n', b );

```

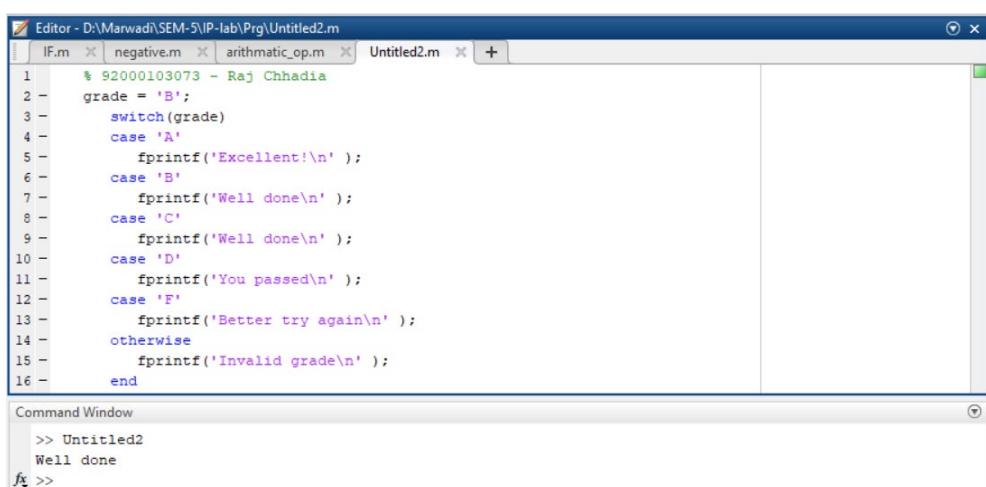
Command Window

```

>> IF
Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200

```

- switch statement



Editor - D:\Marwadi\SEM-5\IP-lab\Prg\Untitled2.m

```

1 % 92000103073 - Raj Chhadia
2 grade = 'B';
3 switch(grade)
4 case 'A'
5 fprintf('Excellent!\n');
6 case 'B'
7 fprintf('Well done\n');
8 case 'C'
9 fprintf('Well done\n');
10 case 'D'
11 fprintf('You passed\n');
12 case 'F'
13 fprintf('Better try again\n');
14 otherwise
15 fprintf('Invalid grade\n');
16 end

```

Command Window

```

>> Untitled2
Well done
fx >>

```

➤ Loop Types in MATLAB

- **while loop**

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\loops.m

```

1 % 92000103073 - Raj Chhadia
2 a = 10;
3 % while loop execution
4 while( a < 20 )
5 fprintf('value of a: %d\n', a);
6 a = a + 1;
7 end

```

Command Window

```

>> loops
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
f1 >>

```

- **for loop**

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\loops.m

```

1 % 92000103073 - Raj Chhadia
2 for a = 10:20
3 fprintf('value of a: %d\n', a);
4 end

```

Command Window

```

>> loops
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
value of a: 20
f1 >>

```

- **nested loops**

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\loops.m

```

1 % 92000103073 - Raj Chhadia
2 for i = 2:10
3 for j = 2:10|
4 if(~mod(i,j))
5 break; % if factor found, not prime
6 end
7 end
8 if(j > (i/j))
9 fprintf('%d is prime\n', i);
10 end
11 end

```

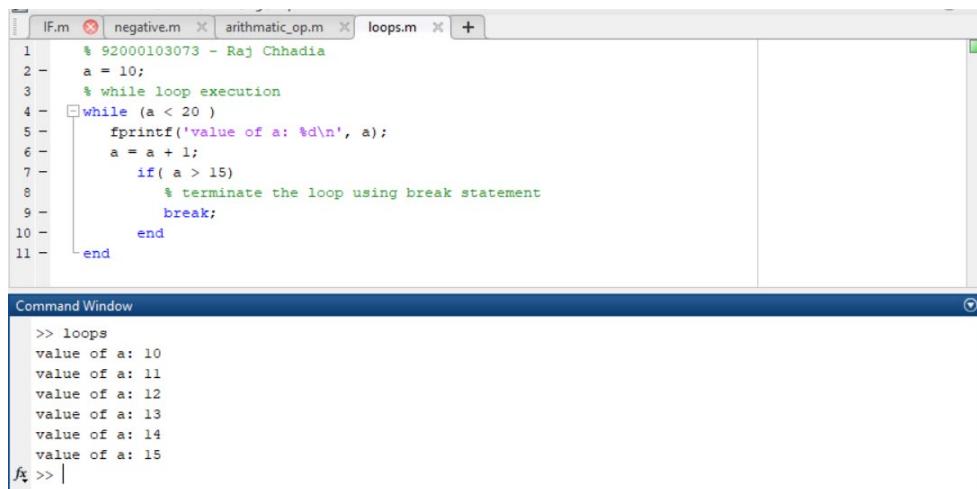
Command Window

```

>> loops
2 is prime
3 is prime
5 is prime
7 is prime
f1 >>

```

- **break statement**



The screenshot shows the MATLAB IDE with the following code in the Editor:

```

1 % 92000103073 - Raj Chhadia
2 a = 10;
3 % while loop execution
4 while (a < 20)
5 fprintf('value of a: %d\n', a);
6 a = a + 1;
7 if( a > 15)
8   % terminate the loop using break statement
9   break;
10 end
11

```

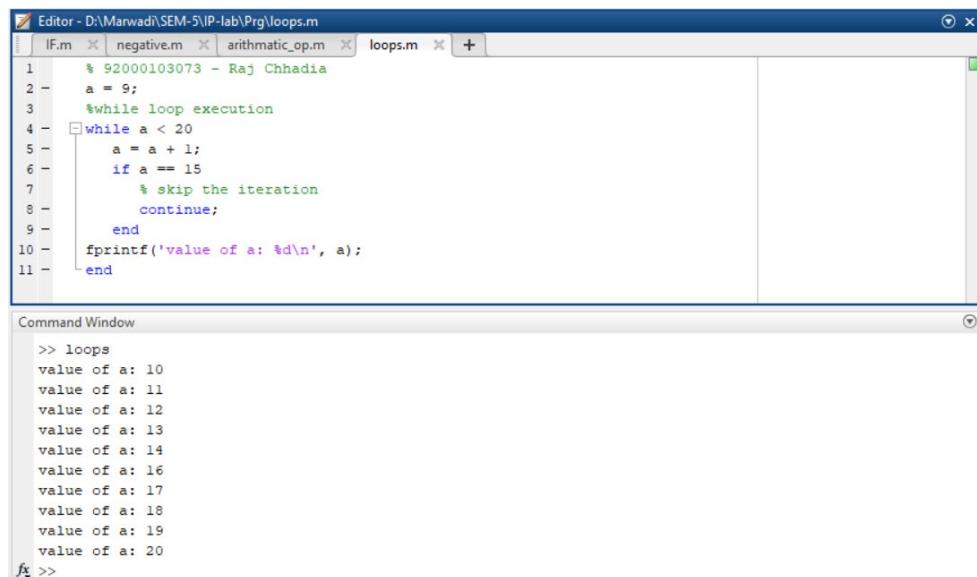
The Command Window output is:

```

>> loops
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
f4 >> |

```

- **continue statement**



The screenshot shows the MATLAB IDE with the following code in the Editor:

```

1 % 92000103073 - Raj Chhadia
2 a = 9;
3 %while loop execution
4 while a < 20
5   a = a + 1;
6   if a == 15
7     % skip the iteration
8     continue;
9   end
10  fprintf('value of a: %d\n', a);
11

```

The Command Window output is:

```

>> loops
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
value of a: 20
f4 >>

```

➤ **Vectors in MATLAB**

- **Row Vectors**



The screenshot shows the MATLAB IDE with the following code in the Editor:

```

1 % 92000103073 - Raj Chhadia
2 r = [7 8 9 10 11]

```

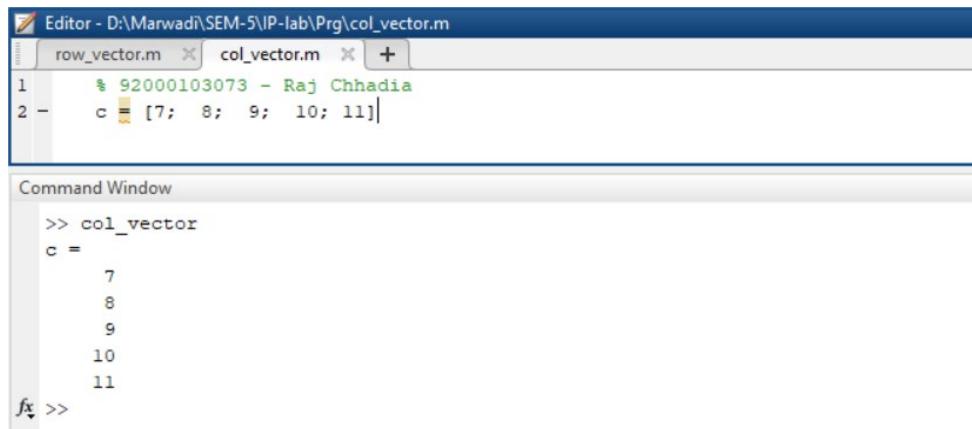
The Command Window output is:

```

>> row_vector
r =
    7     8     9     10    11
f4 >>

```

- Column Vectors



The screenshot shows the MATLAB environment. In the Editor tab, there are two files: 'row_vector.m' and 'col_vector.m'. The 'col_vector.m' file contains the following code:

```

1 % 92000103073 - Raj Chhadia
2 c = [7; 8; 9; 10; 11];

```

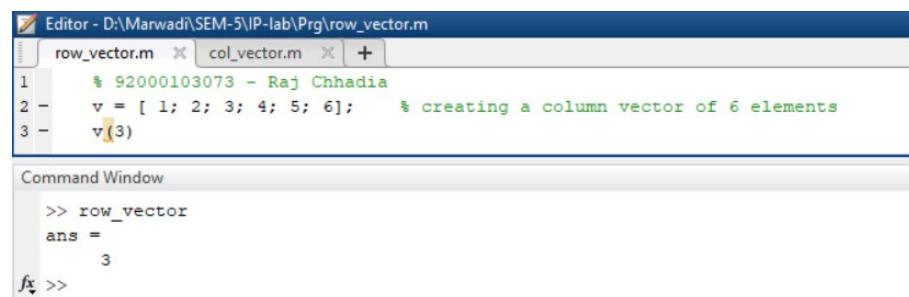
In the Command Window, the user runs the command `>> col_vector`. The output is:

```

>> col_vector
c =
    7
    8
    9
   10
   11

```

- Referencing the Elements of a Vector



The screenshot shows the MATLAB environment. In the Editor tab, there are two files: 'row_vector.m' and 'col_vector.m'. The 'row_vector.m' file contains the following code:

```

1 % 92000103073 - Raj Chhadia
2 v = [1; 2; 3; 4; 5; 6]; % creating a column vector of 6 elements
3 v(3)

```

In the Command Window, the user runs the command `>> row_vector`. The output is:

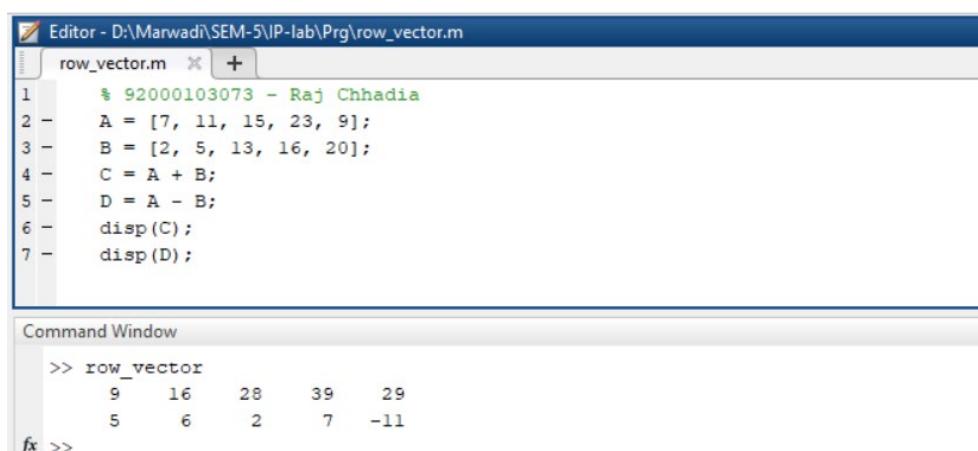
```

>> row_vector
ans =
    3

```

- Vector Operations

- Addition and Subtraction of Vectors



The screenshot shows the MATLAB environment. In the Editor tab, there is one file 'row_vector.m' containing the following code:

```

1 % 92000103073 - Raj Chhadia
2 A = [7, 11, 15, 23, 9];
3 B = [2, 5, 13, 16, 20];
4 C = A + B;
5 D = A - B;
6 disp(C);
7 disp(D);

```

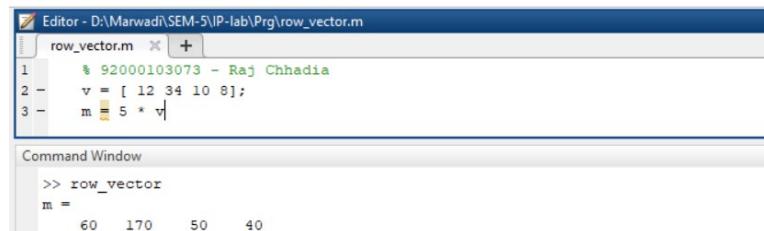
In the Command Window, the user runs the command `>> row_vector`. The output is:

```

>> row_vector
    9      16      28      39      29
    5       6       2       7     -11

```

▪ Scalar Multiplication of Vectors



Editor - D:\Marwadi\SEM-5\IP-lab\Prg\row_vector.m

```

1 % 92000103073 - Raj Chhadia
2 v = [ 12 34 10 8];
3 m = 5 * v;

```

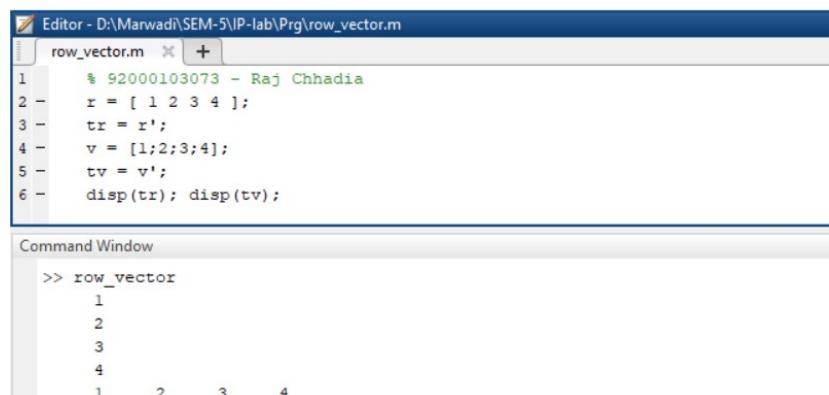
Command Window

```

>> row_vector
m =
     60    170    50    40

```

▪ Transpose of a Vector



Editor - D:\Marwadi\SEM-5\IP-lab\Prg\row_vector.m

```

1 % 92000103073 - Raj Chhadia
2 r = [ 1 2 3 4 ];
3 tr = r';
4 v = [1;2;3;4];
5 tv = v';
6 disp(tr); disp(tv);

```

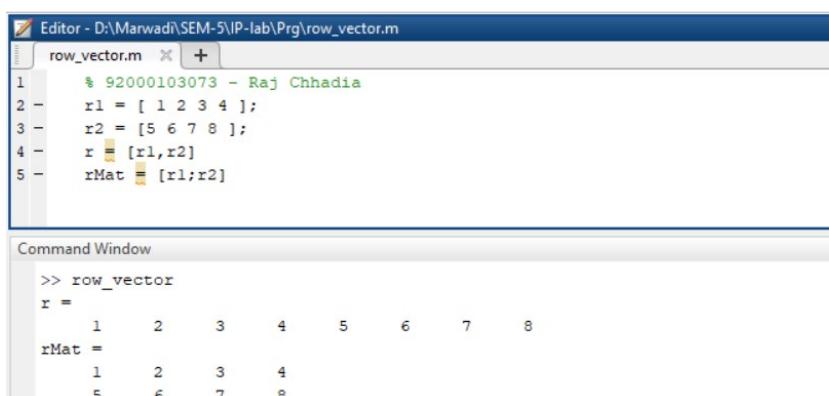
Command Window

```

>> row_vector
1
2
3
4
1    2    3    4

```

▪ Appending Vectors



Editor - D:\Marwadi\SEM-5\IP-lab\Prg\row_vector.m

```

1 % 92000103073 - Raj Chhadia
2 r1 = [ 1 2 3 4 ];
3 r2 = [ 5 6 7 8 ];
4 r = [r1,r2]
5 rMat = [r1;r2]

```

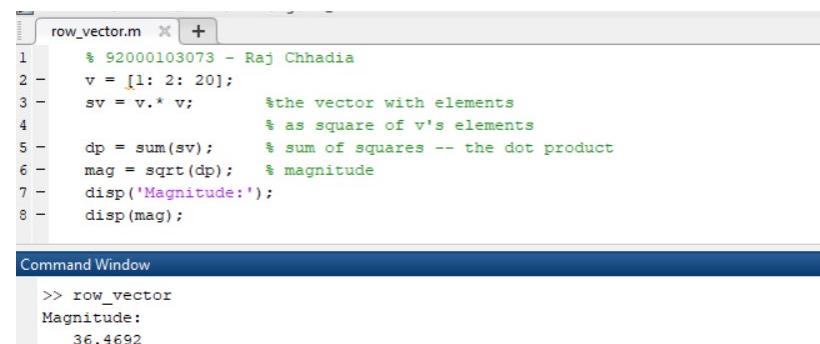
Command Window

```

>> row_vector
r =
     1     2     3     4     5     6     7     8
rMat =
     1     2     3     4
     5     6     7     8

```

▪ Magnitude of a Vector



Editor - D:\Marwadi\SEM-5\IP-lab\Prg\row_vector.m

```

1 % 92000103073 - Raj Chhadia
2 v = [1: 2: 20];
3 sv = v.* v;           %the vector with elements
4 dp = sum(sv);         % sum of squares -- the dot product
5 mag = sqrt(dp);       % magnitude
6 disp('Magnitude:');
7 disp(mag);

```

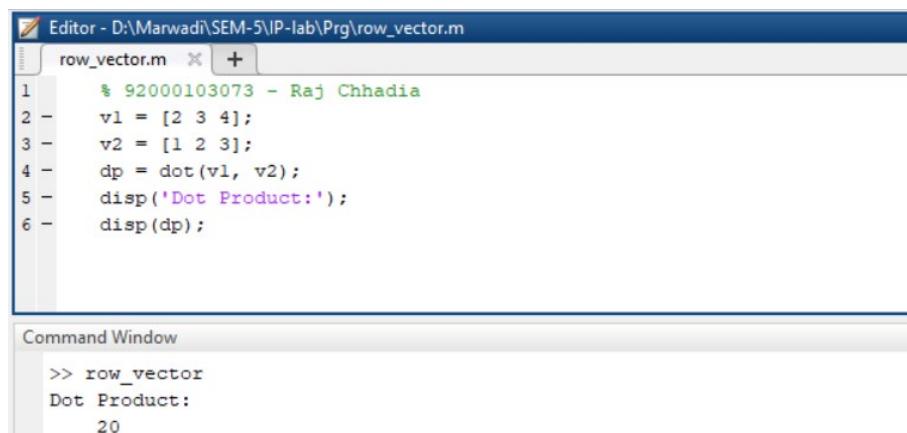
Command Window

```

>> row_vector
Magnitude:
36.4692

```

▪ Vector Dot Product



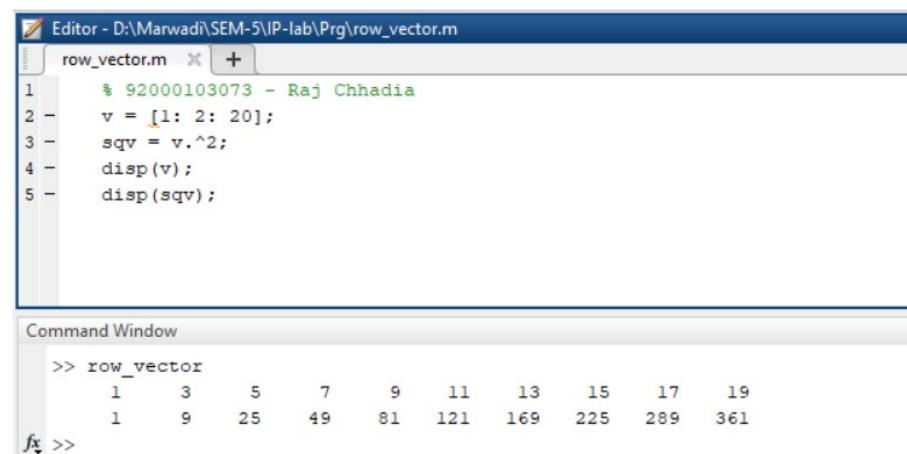
Editor - D:\Marwadi\SEM-5\IP-lab\Prg\row_vector.m

```
row_vector.m +  
1 % 92000103073 - Raj Chhadia  
2 v1 = [2 3 4];  
3 v2 = [1 2 3];  
4 dp = dot(v1, v2);  
5 disp('Dot Product:');  
6 disp(dp);
```

Command Window

```
>> row_vector  
Dot Product:  
20
```

▪ Vectors with Uniformly Spaced Elements



Editor - D:\Marwadi\SEM-5\IP-lab\Prg\row_vector.m

```
row_vector.m +  
1 % 92000103073 - Raj Chhadia  
2 v = [1: 2: 20];  
3 sqv = v.^2;  
4 disp(v);  
5 disp(sqv);
```

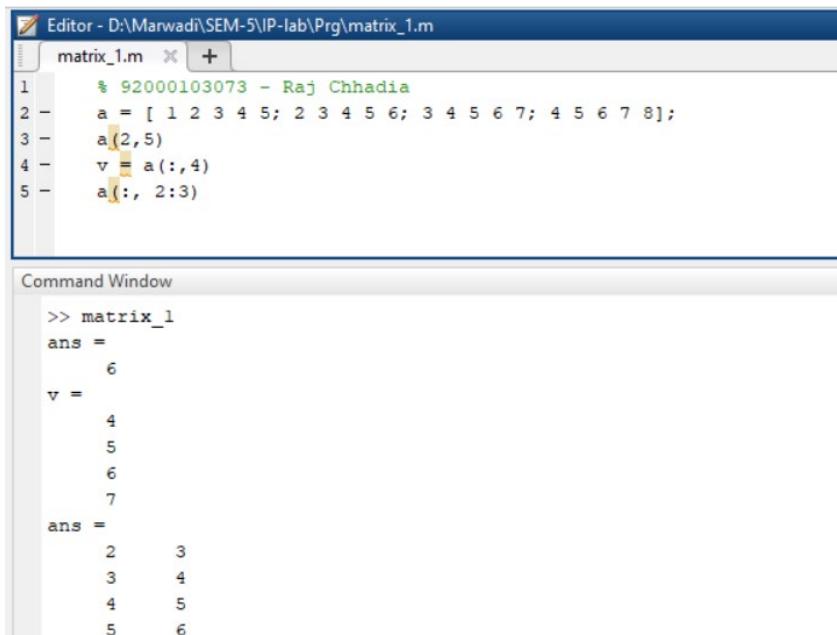
Command Window

```
>> row_vector  
1 3 5 7 9 11 13 15 17 19  
1 9 25 49 81 121 169 225 289 361  
fx >>
```

➤ Matrix in MATLAB

○ Referencing the Elements of a Matrix

- To reference an element in the m^{th} row and n^{th} column, of a matrix mx , we write –
 $mx(m, n);$
- To reference all the elements in the m^{th} column we type $A(:, m)$.



The screenshot shows the MATLAB environment. In the Editor window, a script named 'matrix_1.m' is open with the following code:

```

1 % 92000103073 - Raj Chhadia
2 a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
3 a(2,5)
4 v = a(:,4)
5 a(:, 2:3)

```

In the Command Window, the output is:

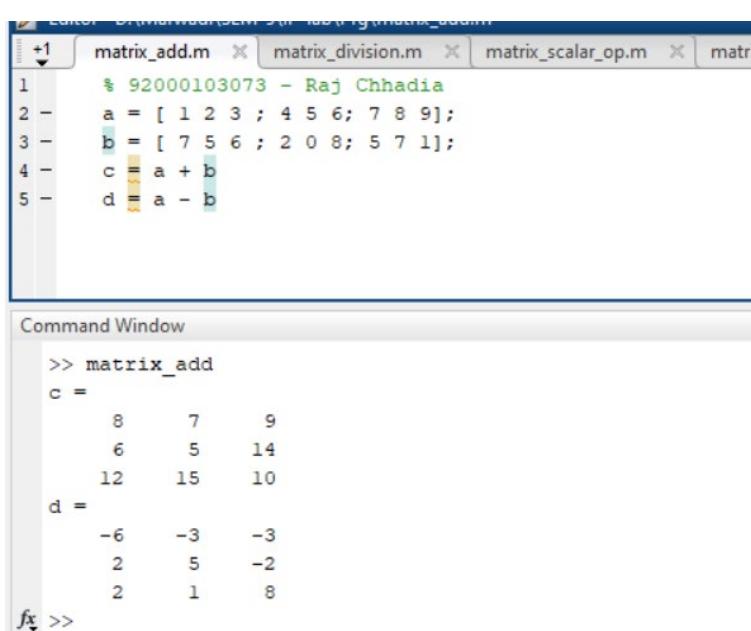
```

>> matrix_1
ans =
     6
v =
     4
     5
     6
     7
ans =
     2     3
     3     4
     4     5
     5     6

```

○ Matrix Operations

■ Addition and Subtraction of Matrices



The screenshot shows the MATLAB environment. In the Editor window, a script named 'matrix_add.m' is open with the following code:

```

1 % 92000103073 - Raj Chhadia
2 a = [ 1 2 3 ; 4 5 6; 7 8 9];
3 b = [ 7 5 6 ; 2 0 8; 5 7 1];
4 c = a + b
5 d = a - b

```

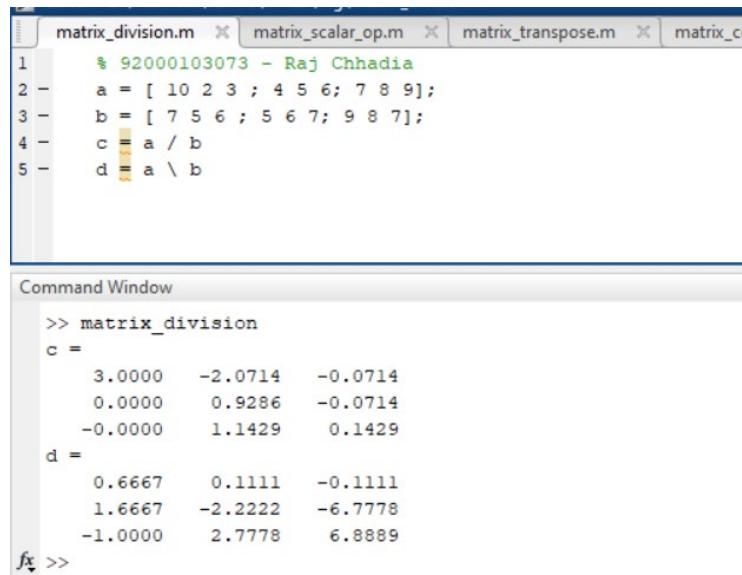
In the Command Window, the output is:

```

>> matrix_add
c =
     8      7      9
     6      5     14
    12     15     10
d =
    -6     -3     -3
     2      5     -2
     2      1      8

```

■ Division of Matrices



```

matrix_division.m
1 % 92000103073 - Raj Chhadia
2 a = [ 10 2 3 ; 4 5 6; 7 8 9];
3 b = [ 7 5 6 ; 5 6 7; 9 8 7];
4 c = a / b
5 d = a \ b

```

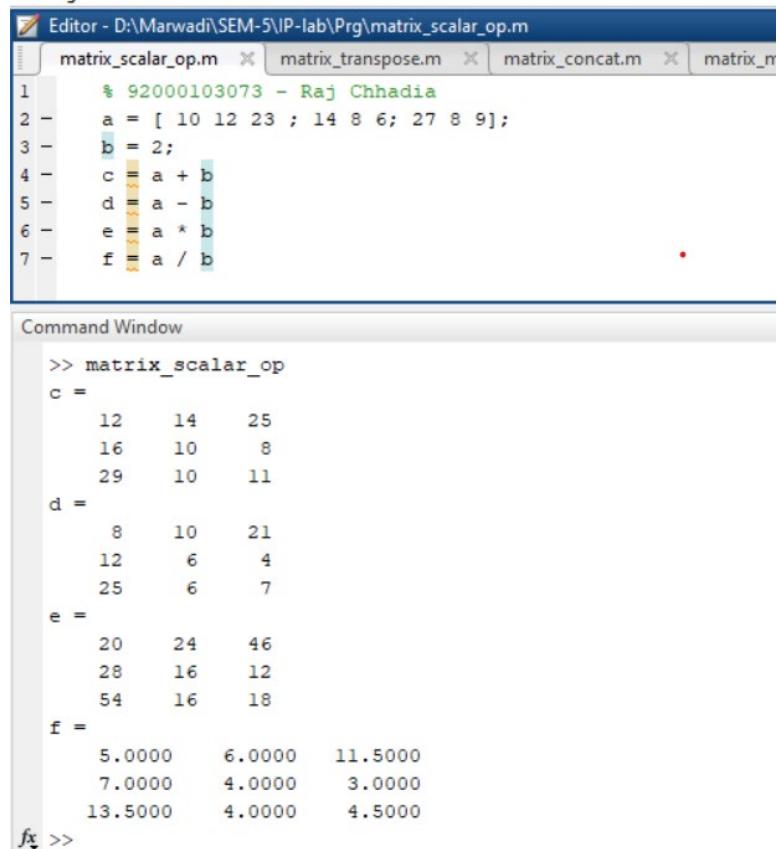
Command Window

```

>> matrix_division
c =
    3.0000   -2.0714   -0.0714
    0.0000    0.9286   -0.0714
   -0.0000    1.1429    0.1429
d =
    0.6667    0.1111   -0.1111
    1.6667   -2.2222   -6.7778
   -1.0000    2.7778    6.8889
f >>

```

■ Scalar Operations of Matrices



```

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\matrix_scalar_op.m
matrix_scalar_op.m
1 % 92000103073 - Raj Chhadia
2 a = [ 10 12 23 ; 14 8 6; 27 8 9];
3 b = 2;
4 c = a + b
5 d = a - b
6 e = a * b
7 f = a / b

```

Command Window

```

>> matrix_scalar_op
c =
    12    14    25
    16    10     8
    29    10    11
d =
     8    10    21
    12     6     4
    25     6     7
e =
    20    24    46
    28    16    12
    54    16    18
f =
    5.0000    6.0000   11.5000
    7.0000    4.0000    3.0000
   13.5000    4.0000    4.5000
f >>

```

▪ Transpose of a Matrix

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\matrix_transpose.m

```

matrix_transpose.m  matrix_concat.m  matrix_multiplicat
1 % 92000103073 - Raj Chhadia
2 - a = [ 10 12 23 ; 14 8 6; 27 8 9]
3 - b = a'

```

Command Window

```

>> matrix_transpose
a =
    10     12     23
    14      8      6
    27      8      9
b =
    10     14     27
    12      8      8
    23      6      9
fx >>

```

▪ Concatenating Matrices

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\matrix_concat.m

```

matrix_concat.m  matrix_multiplication.m  matrix_determinant.m
1 % 92000103073 - Raj Chhadia
2 - a = [ 10 12 23 ; 14 8 6; 27 8 9]
3 - b = [ 12 31 45 ; 8 0 -9; 45 2 11]
4 - c = [a, b]
5 - d = [a; b]

```

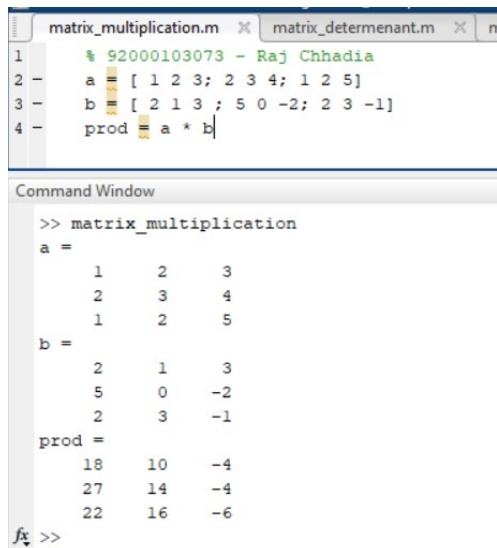
Command Window

```

>> matrix_concat
a =
    10     12     23
    14      8      6
    27      8      9
b =
    12     31     45
     8      0     -9
    45      2     11
c =
    10     12     23     12     31     45
    14      8      6      8      0     -9
    27      8      9     45      2     11
d =
    10     12     23
    14      8      6
    27      8      9
    12     31     45
     8      0     -9
    45      2     11
fx >>

```

■ Matrix Multiplication

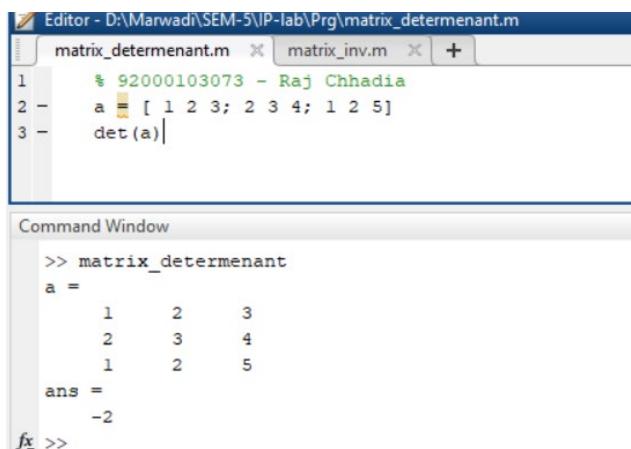


```

matrix_multiplication.m
1 % 92000103073 - Raj Chhadia
2 a = [ 1 2 3; 2 3 4; 1 2 5]
3 b = [ 2 1 3 ; 5 0 -2; 2 3 -1]
4 prod = a * b

Command Window
>> matrix_multiplication
a =
     1     2     3
     2     3     4
     1     2     5
b =
     2     1     3
     5     0    -2
     2     3    -1
prod =
     18    10    -4
    27    14    -4
    22    16    -6
f1 >>
  
```

■ Determinant of a Matrix

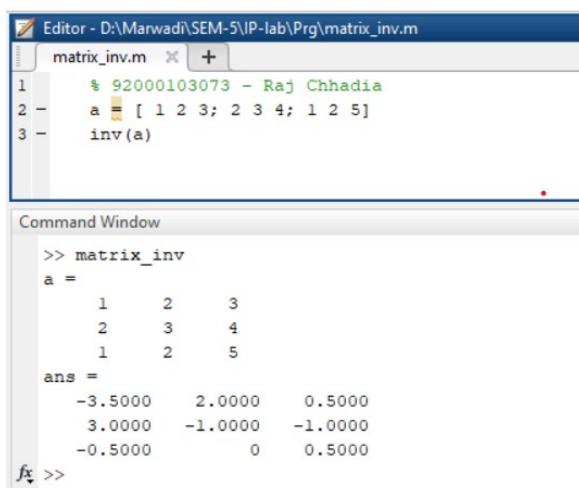


```

matrix_determinant.m
1 % 92000103073 - Raj Chhadia
2 a = [ 1 2 3; 2 3 4; 1 2 5]
3 det(a)

Command Window
>> matrix_determinant
a =
     1     2     3
     2     3     4
     1     2     5
ans =
    -2
f1 >>
  
```

■ Inverse of a Matrix



```

matrix_inv.m
1 % 92000103073 - Raj Chhadia
2 a = [ 1 2 3; 2 3 4; 1 2 5]
3 inv(a)

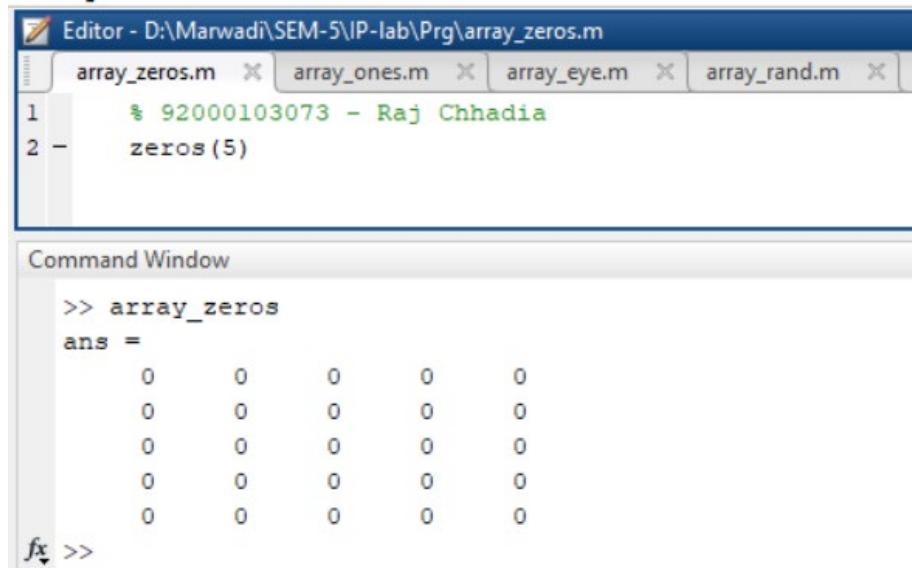
Command Window
>> matrix_inv
a =
     1     2     3
     2     3     4
     1     2     5
ans =
    -3.5000    2.0000    0.5000
     3.0000   -1.0000   -1.0000
    -0.5000        0    0.5000
f1 >>
  
```

➤ Arrays in MATLAB

○ Special Arrays in MATLAB

■ zeros() function

→ The zeros() function creates an array of all zeros .



The screenshot shows the MATLAB environment. The Editor window at the top contains the code:

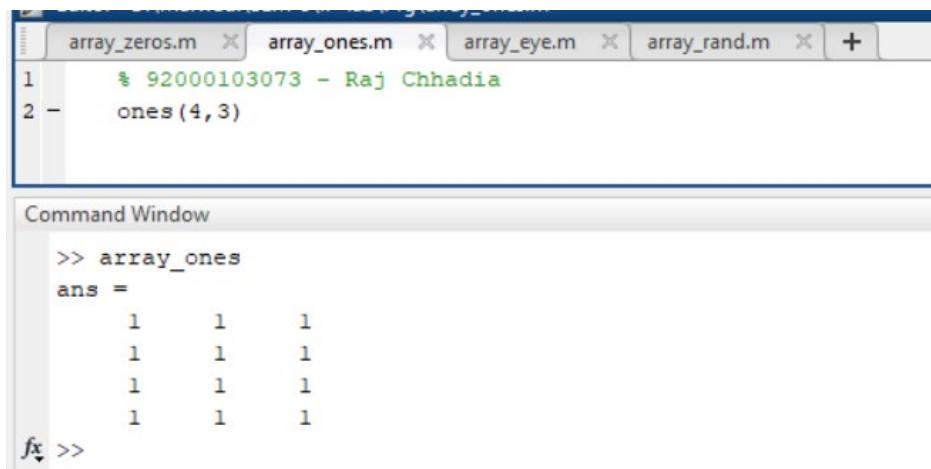
```
Editor - D:\Marwadi\SEM-5\IP-lab\Prg\array_zeros.m
array_zeros.m x array_ones.m x array_eye.m x array_rand.m x
1 % 92000103073 - Raj Chhadia
2 - zeros(5)
```

The Command Window below shows the result of running the code:

```
Command Window
>> array_zeros
ans =
    0     0     0     0     0
    0     0     0     0     0
    0     0     0     0     0
    0     0     0     0     0
    0     0     0     0     0
fx >>
```

■ ones() function

→ The ones() function creates an array of all ones.



The screenshot shows the MATLAB environment. The Editor window at the top contains the code:

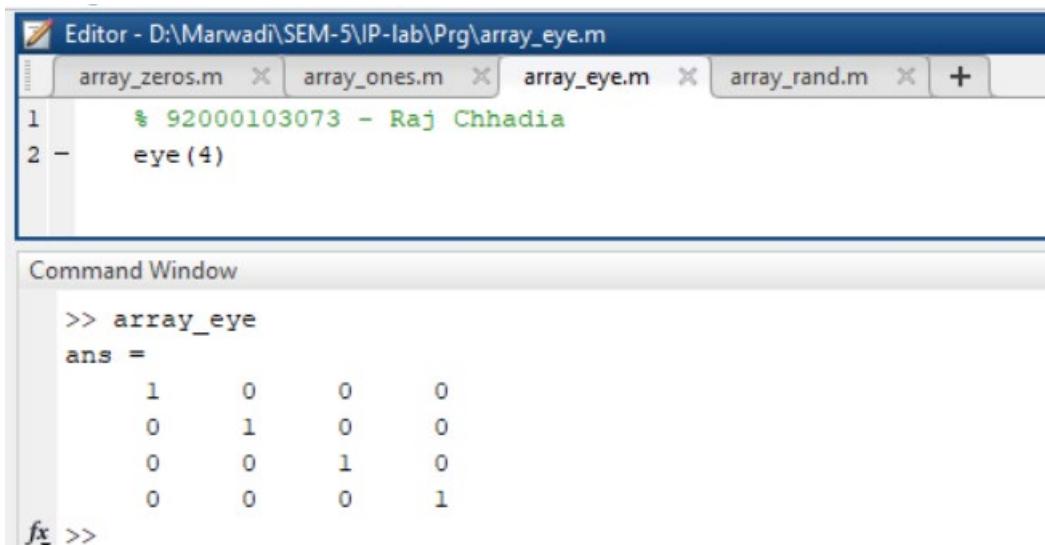
```
Editor - D:\Marwadi\SEM-5\IP-lab\Prg\array_zeros.m x array_ones.m x array_eye.m x array_rand.m x +
1 % 92000103073 - Raj Chhadia
2 - ones(4,3)
```

The Command Window below shows the result of running the code:

```
Command Window
>> array_ones
ans =
    1     1     1
    1     1     1
    1     1     1
    1     1     1
fx >>
```

- **eye() function**

→ The eye() function creates an identity matrix.



The screenshot shows the MATLAB environment. The Editor window displays the code:

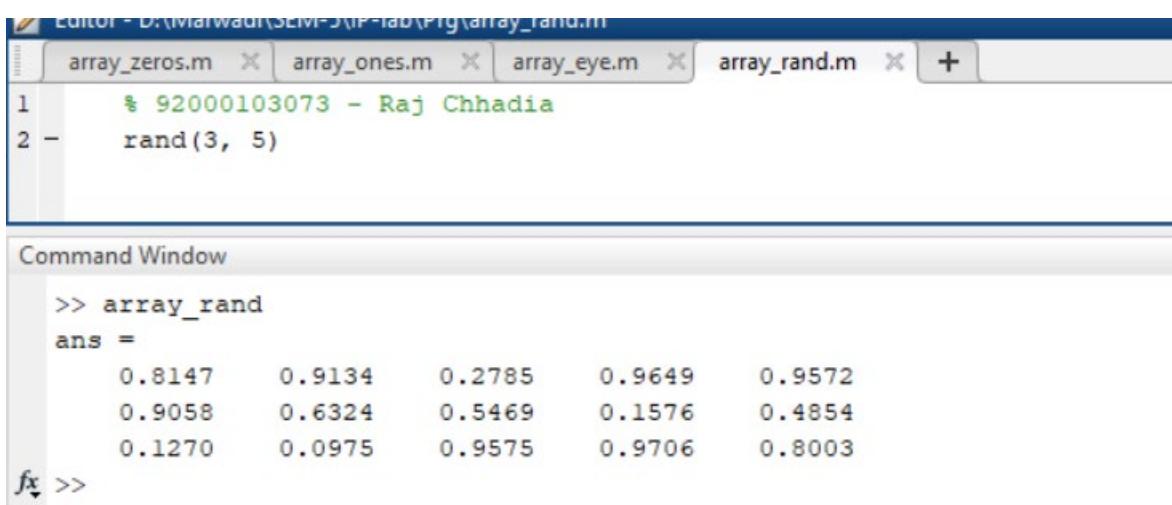
```
% 92000103073 - Raj Chhadia
eye(4)
```

The Command Window displays the output:

```
>> array_eye
ans =
  1   0   0   0
  0   1   0   0
  0   0   1   0
  0   0   0   1
```

- **rand() function**

→ The rand() function creates an array of uniformly distributed random numbers on (0,1) .



The screenshot shows the MATLAB environment. The Editor window displays the code:

```
% 92000103073 - Raj Chhadia
rand(3, 5)
```

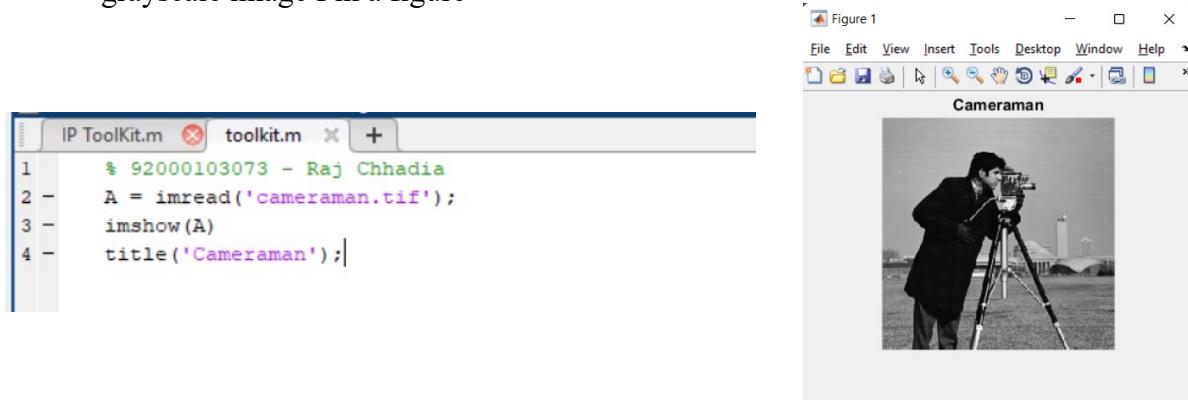
The Command Window displays the output:

```
>> array_rand
ans =
  0.8147    0.9134    0.2785    0.9649    0.9572
  0.9058    0.6324    0.5469    0.1576    0.4854
  0.1270    0.0975    0.9575    0.9706    0.8003
```

➤ Image Processing Toolkit

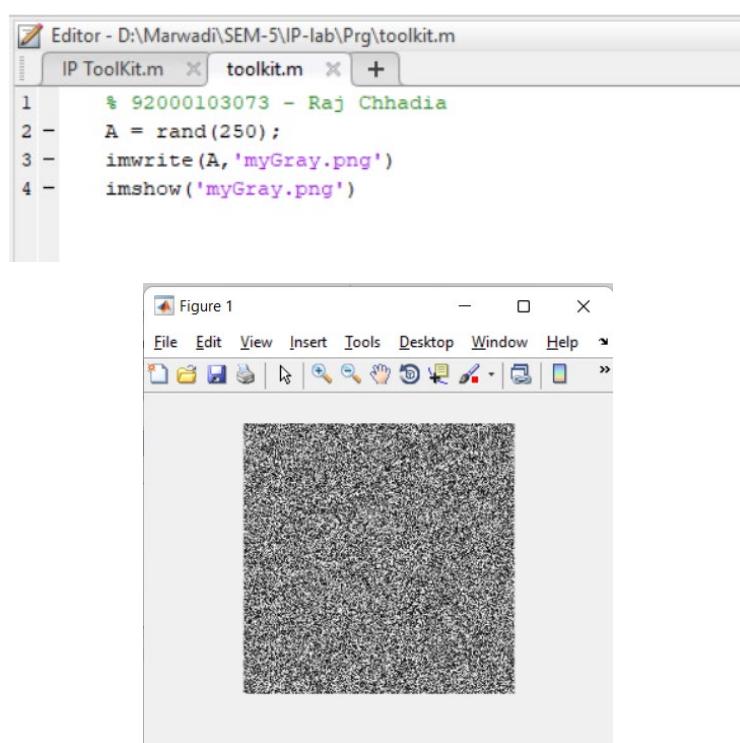
○ imread() and imshow() command

→ imread() reads the image from the file specified by filename, imshow() displays the grayscale image I in a figure



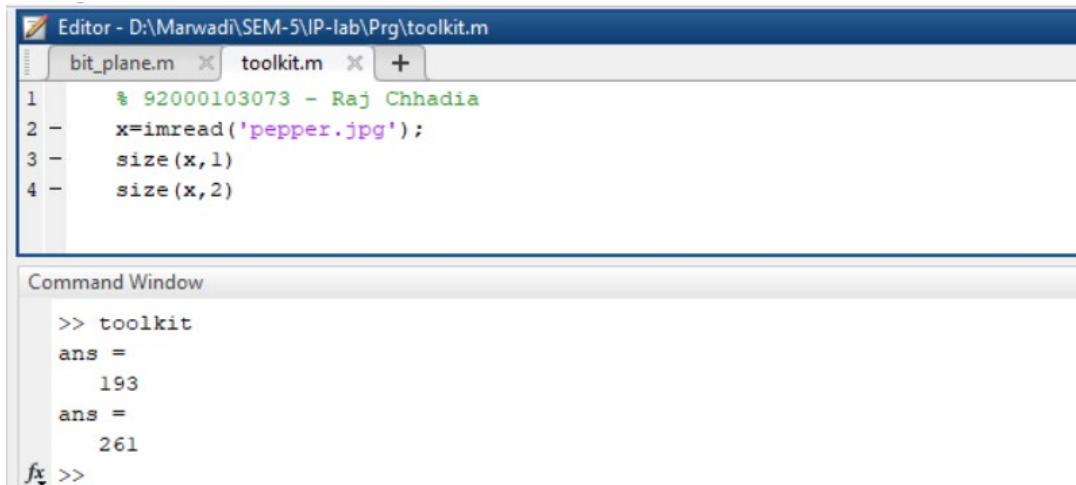
○ imwrite() command

→ imwrite() writes image data A to the file specified by filename.



- **size() command**

→ size() is used to know number of pixels in a image.



The screenshot shows the MATLAB environment. The Editor window at the top contains the following code:

```

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\toolkit.m
bit_plane.m toolkit.m +
1 % 92000103073 - Raj Chhadia
2 x=imread('pepper.jpg');
3 size(x,1)
4 size(x,2)

```

The Command Window below shows the output of the code:

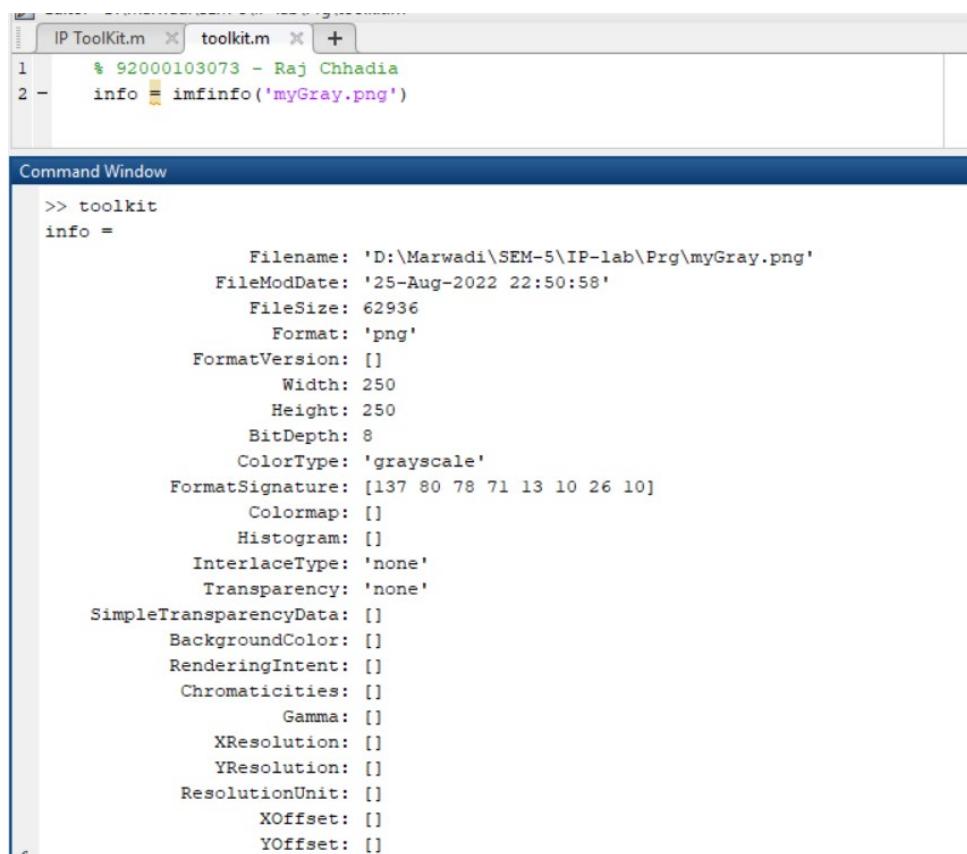
```

Command Window
>> toolkit
ans =
 193
ans =
 261
fx >>

```

- **imfinfo() command**

→ imfinfo() returns a structure whose fields contain information about an image in a graphics file.



The screenshot shows the MATLAB environment. The Editor window at the top contains the following code:

```

IP ToolKit.m toolkit.m +
1 % 92000103073 - Raj Chhadia
2 info = imfinfo('myGray.png')

```

The Command Window below shows the output of the code, displaying a detailed structure of the image file information:

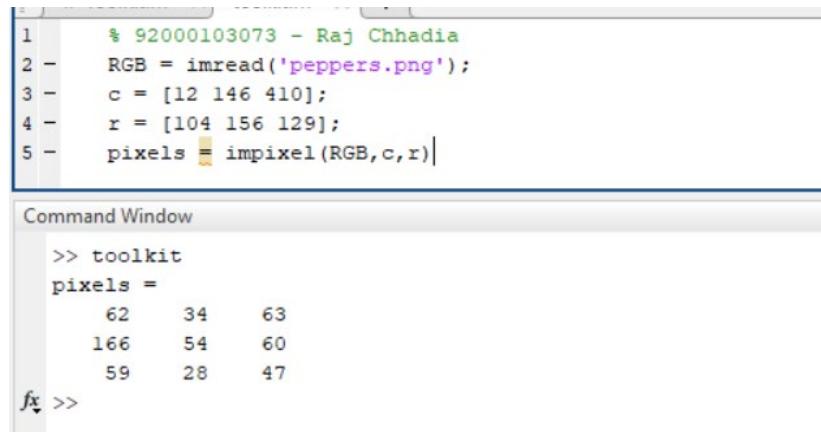
```

Command Window
>> toolkit
info =
  struct with fields:
    Filename: 'D:\Marwadi\SEM-5\IP-lab\Prg\myGray.png'
    FileModDate: '25-Aug-2022 22:50:58'
    FileSize: 62936
    Format: 'png'
    FormatVersion: []
    Width: 250
    Height: 250
    BitDepth: 8
    ColorType: 'grayscale'
    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: []

```

- **impixel() command**

→ `impixel()` lets you select pixels interactively from the image in the current axes.



```

1 % 92000103073 - Raj Chhadia
2 - RGB = imread('peppers.png');
3 - c = [12 146 410];
4 - r = [104 156 129];
5 - pixels = impixel(RGB,c,r)

```

Command Window

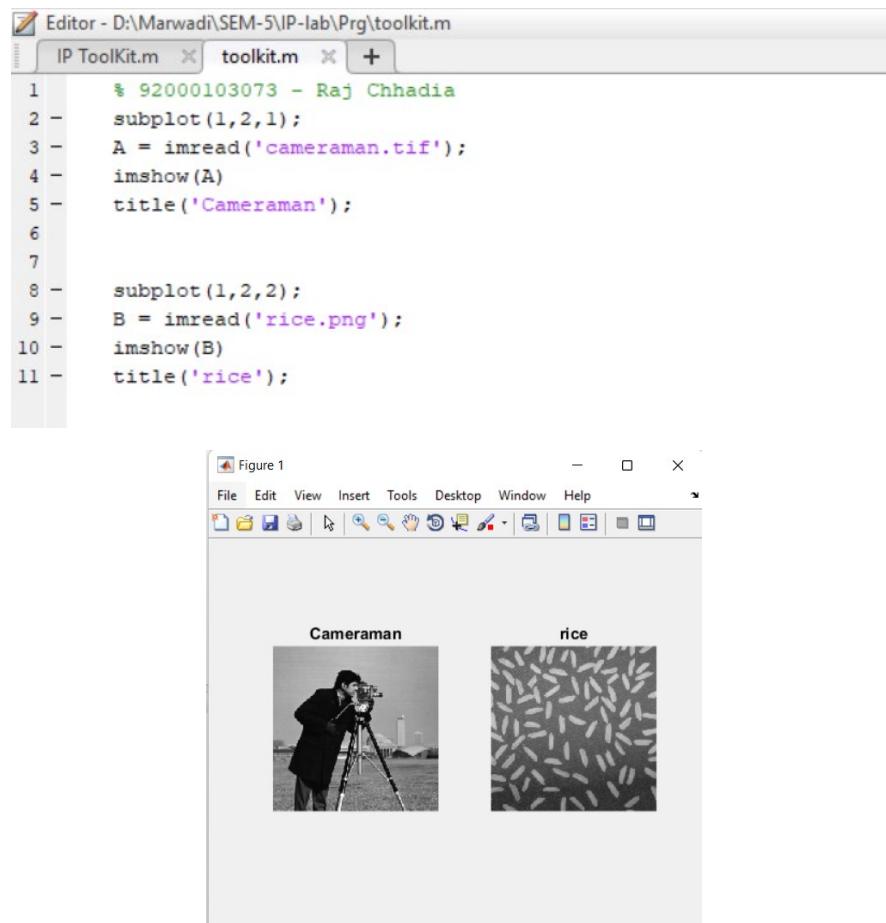
```

>> toolkit
pixels =
    62     34     63
   166     54     60
    59     28     47
fx >>

```

- **subplot() command**

→ `subplot(m,n,p)` divides the current figure into an m-by-n grid and creates axes in the position specified by p.



Editor - D:\Marwadi\SEM-5\IP-lab\Prg\toolkit.m

```

IP ToolKit.m  toolkit.m  +
1 % 92000103073 - Raj Chhadia
2 - subplot(1,2,1);
3 - A = imread('cameraman.tif');
4 - imshow(A)
5 - title('Cameraman');
6
7
8 - subplot(1,2,2);
9 - B = imread('rice.png');
10 - imshow(B)
11 - title('rice');

```

Figure 1

File Edit View Insert Tools Desktop Window Help

Cameraman rice

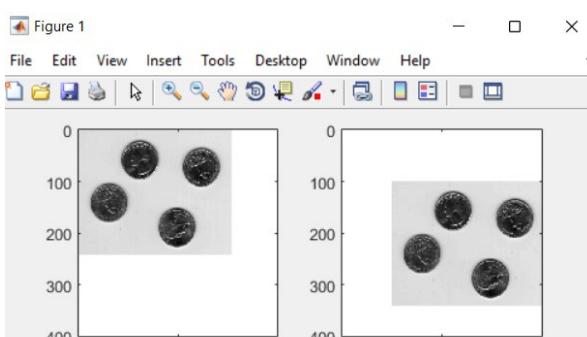
- **imagesc() command**

→ `imagesc(C)` displays the data in array C as an image that uses the full range of colors in the colormap.

```

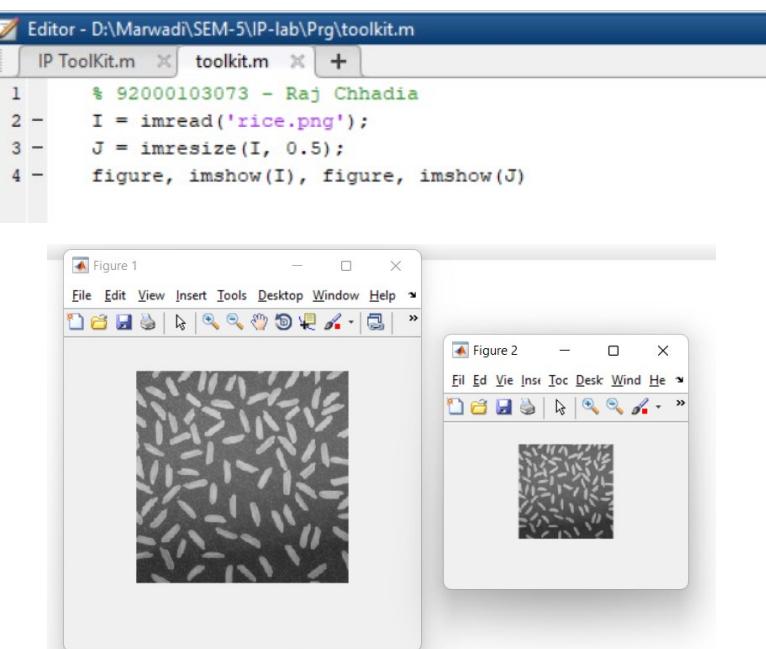
1 % 92000103073 - Raj Chhadia
2 i = imread('eight.tif');
3 figure;
4 subplot(2,2,1);
5 imagesc(i);
6 axis([0 400 0 400]);
7 colormap(gray);
8 subplot(2,2,2);
9 imagesc(100,100,i);
10 axis([0 400 0 400]);
11 colormap(gray);

```



- **imresize() command**

→ `B = imresize(A,scale)` returns image B that is scale times the size of image A.



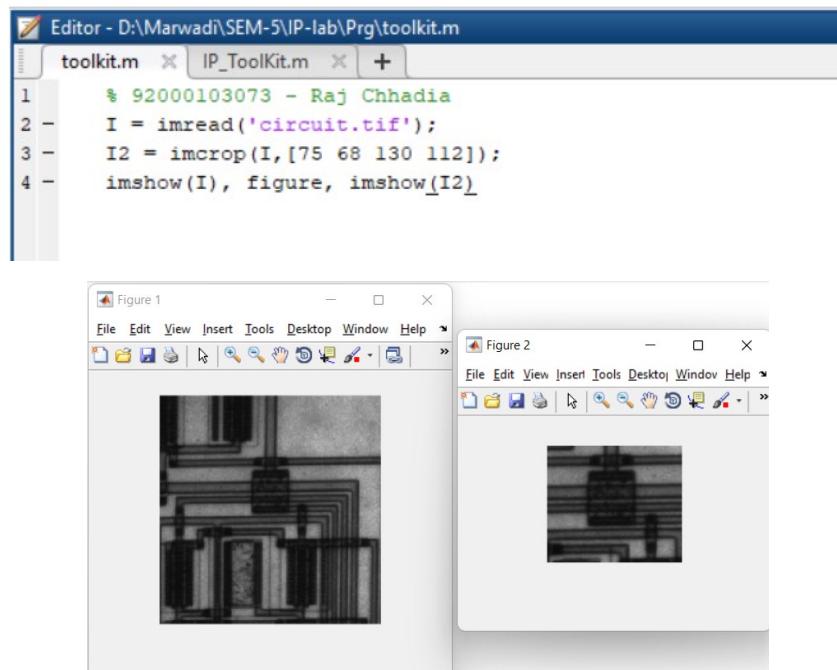
```

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\toolkit.m
  IP ToolKit.m  toolkit.m  +
1 % 92000103073 - Raj Chhadia
2 I = imread('rice.png');
3 J = imresize(I, 0.5);
4 figure, imshow(I), figure, imshow(J)

```

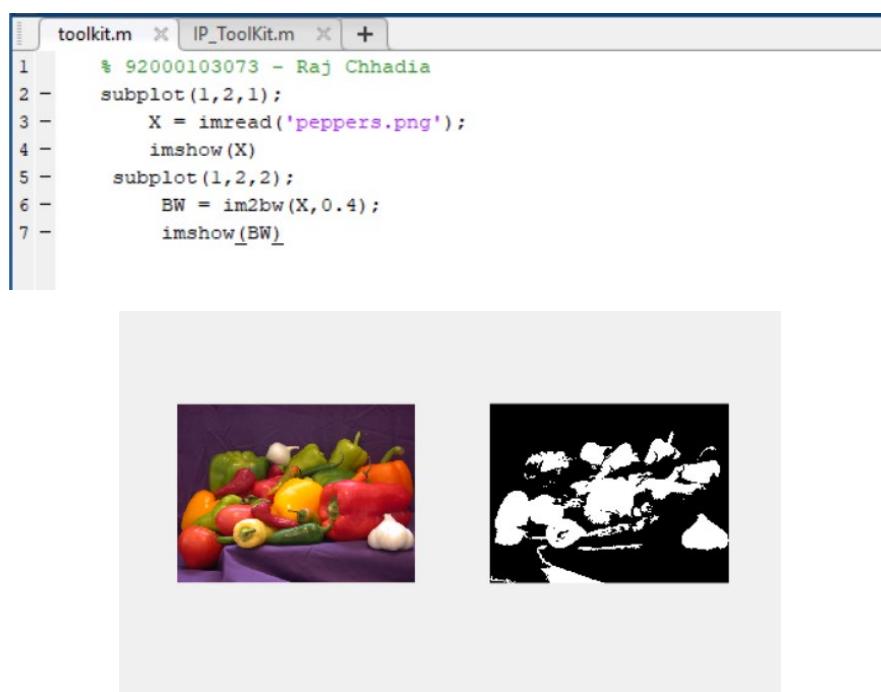
- **imcrop() command**

→ imcrop() creates an interactive Crop Image tool associated with the grayscale, truecolor, or binary image displayed in the current figure.



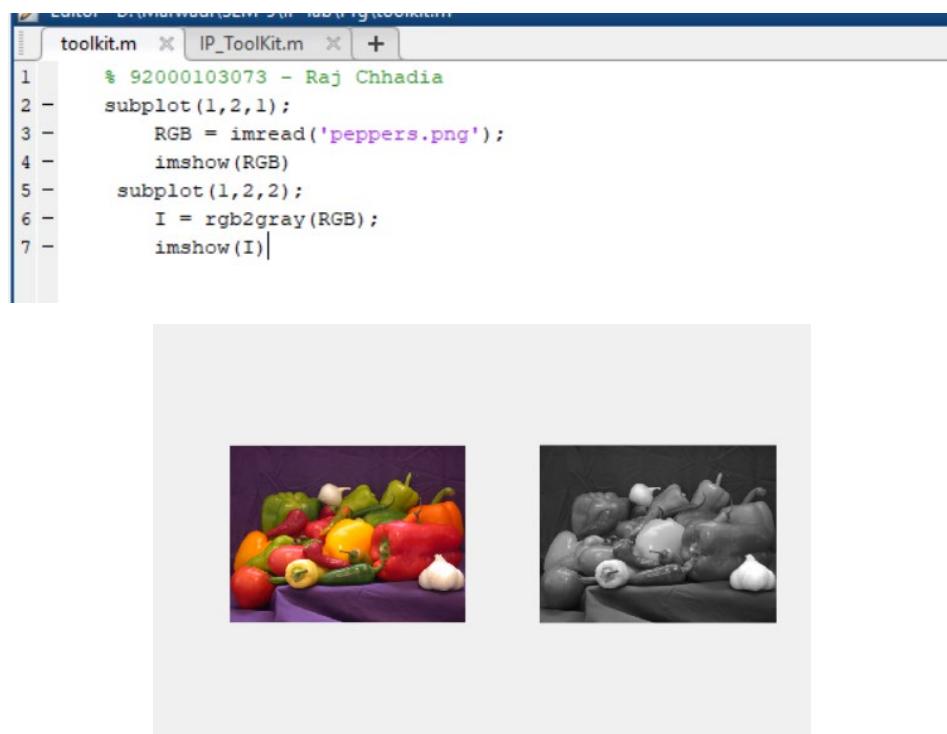
- **im2bw() command**

→ im2bw() converts the grayscale image I to binary image.



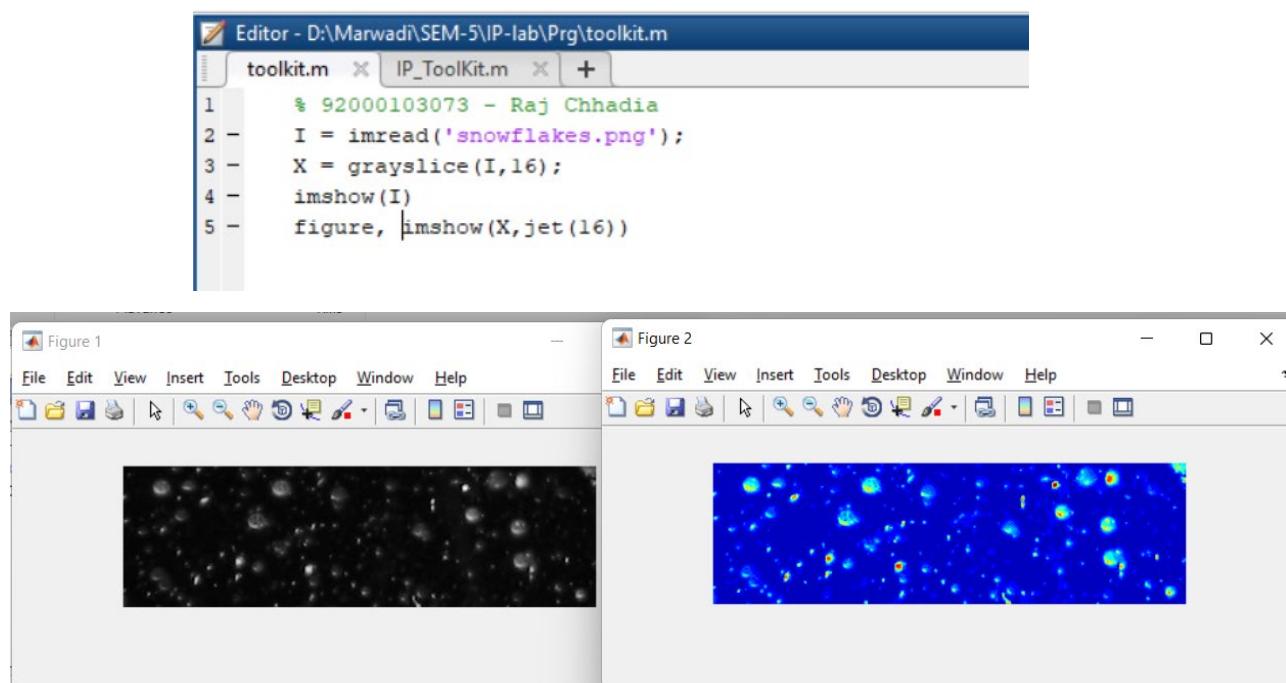
- **rgb2gray() command**

→ The `rgb2gray()` function converts RGB images to grayscale.



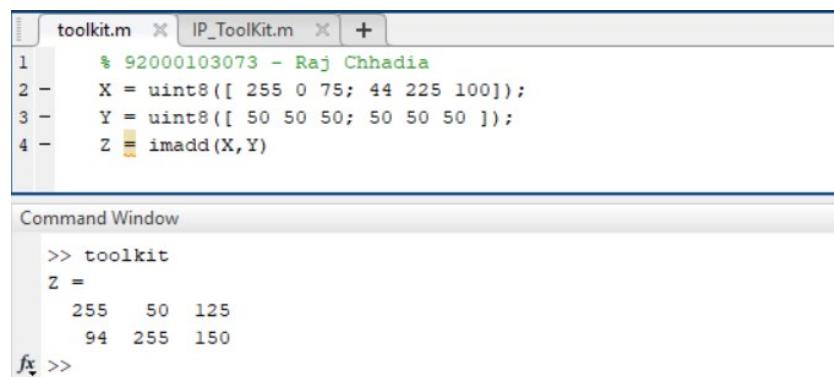
- **grayslice() command**

→ `grayslice()` converts a grayscale image to an indexed image by using multilevel thresholding approach.



- **imadd() command**

→ imadd() adds each element in array X with the corresponding element in array Y and returns the sum.



The screenshot shows the MATLAB IDE. The code editor window contains the following script:

```

1 % 92000103073 - Raj Chhadia
2 - X = uint8([ 255 0 75; 44 225 100]);
3 - Y = uint8([ 50 50 50; 50 50 50 ]);
4 - Z = imadd(X,Y)

```

The Command Window below shows the output of the script:

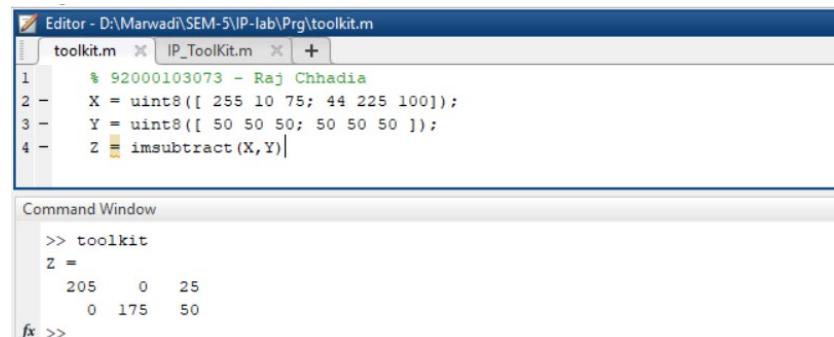
```

>> toolkit
Z =
255 50 125
94 255 150
fx >>

```

- **imssubtract() command**

→ imsubtract() subtracts each element in array Y from the corresponding element in array X and returns the difference.



The screenshot shows the MATLAB IDE. The code editor window contains the following script:

```

1 % 92000103073 - Raj Chhadia
2 - X = uint8([ 255 10 75; 44 225 100]);
3 - Y = uint8([ 50 50 50; 50 50 50 ]);
4 - Z = imsubtract(X,Y)

```

The Command Window below shows the output of the script:

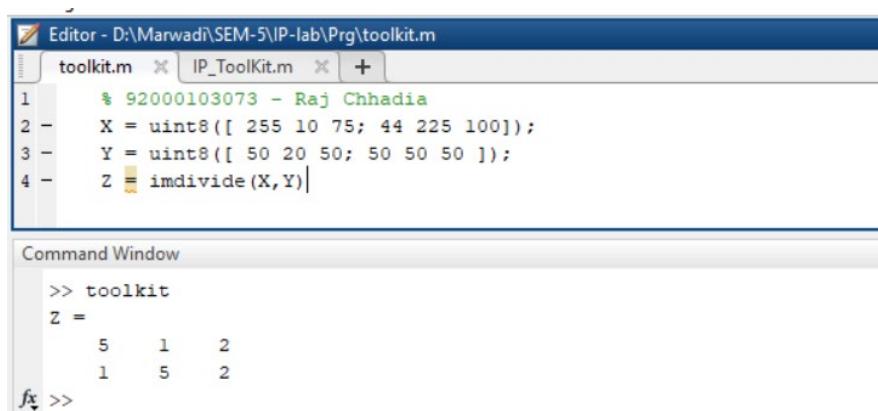
```

>> toolkit
Z =
205 0 25
0 175 50
fx >>

```

- **imdivide() command**

→ imdivide() divides each element in the array X by corresponding element in array Y.



The screenshot shows the MATLAB IDE. The code editor window contains the following script:

```

1 % 92000103073 - Raj Chhadia
2 - X = uint8([ 255 10 75; 44 225 100]);
3 - Y = uint8([ 50 20 50; 50 50 50 ]);
4 - Z = imdivide(X,Y)

```

The Command Window below shows the output of the script:

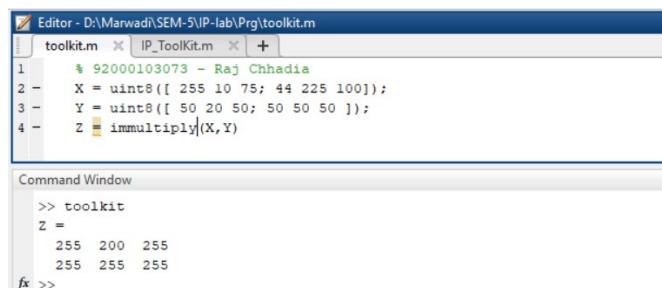
```

>> toolkit
Z =
5 1 2
1 5 2
fx >>

```

- **immultiply() command**

→ immultiply() multiplies each element in array X by the corresponding element in array Y and returns product.



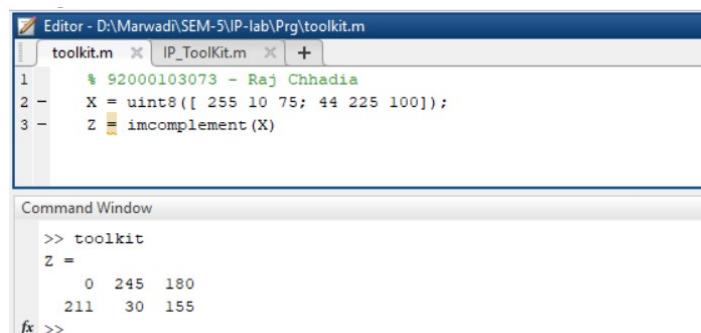
The screenshot shows the MATLAB Editor and Command Window. In the Editor, the code `Z = immultiply(X,Y)` is being typed into a file named toolkit.m. In the Command Window, the command `>> toolkit` is run, followed by `Z =`, which displays the resulting matrix:

```

>> toolkit
Z =
  255 200 255
  255 255 255
  
```

- **imcomplement() command**

→ imcomplement() computes the complement of the image I.



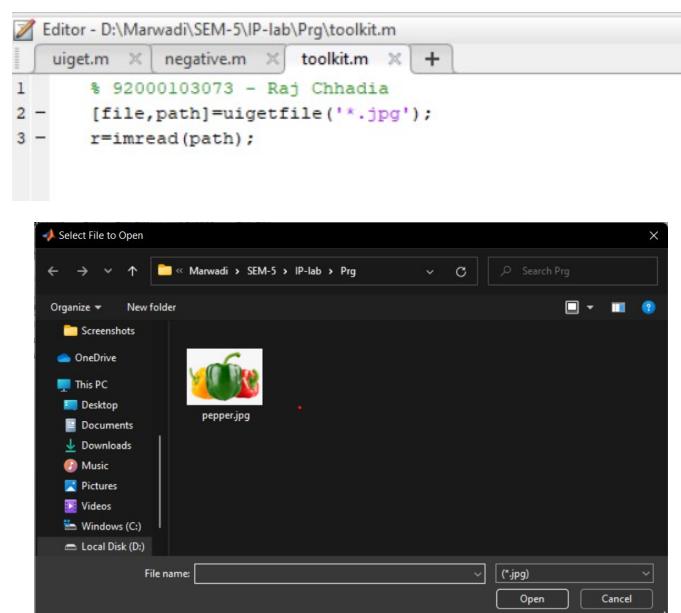
The screenshot shows the MATLAB Editor and Command Window. In the Editor, the code `Z = imcomplement(X)` is being typed into a file named toolkit.m. In the Command Window, the command `>> toolkit` is run, followed by `Z =`, which displays the resulting matrix:

```

>> toolkit
Z =
  0 245 180
  211 30 155
  
```

- **uigetfile() command**

→ uigetfile() opens file selection dialog box.



The screenshot shows the MATLAB Editor and a 'Select File to Open' dialog box. In the Editor, the code `r=imread(path);` is part of a script. The dialog box shows a file named 'pepper.jpg' in the current directory. The file path is displayed as:

```

C:\Marwadi\SEM-5\IP-lab\Prg\pepper.jpg
  
```

Practical 2

Aim: Point processing in spatial domain

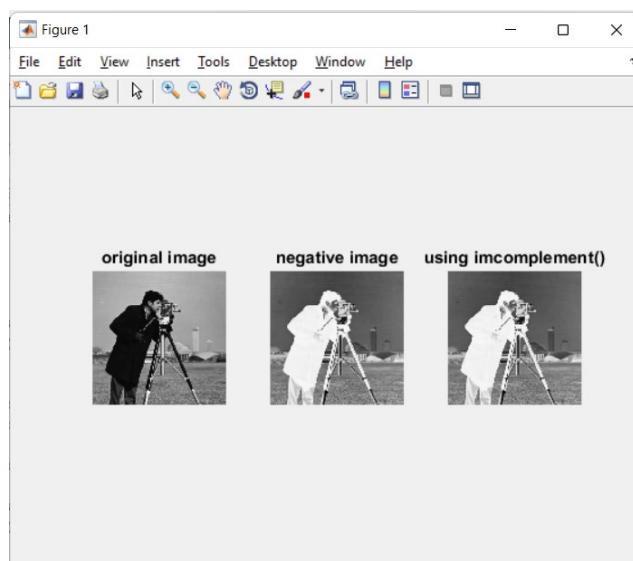
- a. Negation of an image
- b. Thresholding of an image
- c. Contrast Stretching of an image

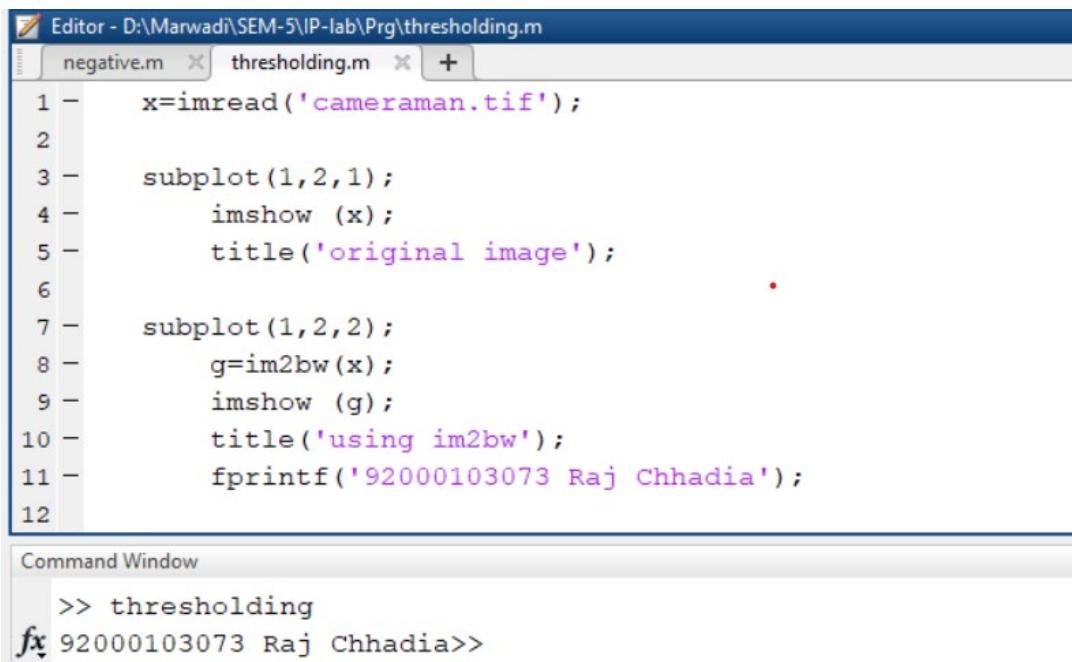
a. Negation of an image

Code:

```
Editor - D:\Marwadi\SEM-5\IP-lab\Prg\negative.m
negative.m  X  +
1 - r = imread('cameraman.tif');
2 - s = r;
3 -
4 - subplot(1,3,1);
5 - imshow(r);
6 - title('original image');
7 -
8 - subplot(1,3,2);
9 - for row = 1 : size(r,1)
10 -   for col = 1 : size(r,2)
11 -     s(row,col) = 255 - r(row,col);
12 -   end
13 - end
14 - imshow(s);
15 - title('negative image');
16 -
17 - subplot(1,3,3);
18 - q = imcomplement(r);
19 - imshow(q);
20 - title('using imcomplement()');
21 - fprintf('92000103073 Raj Chhadia');
```

Output:



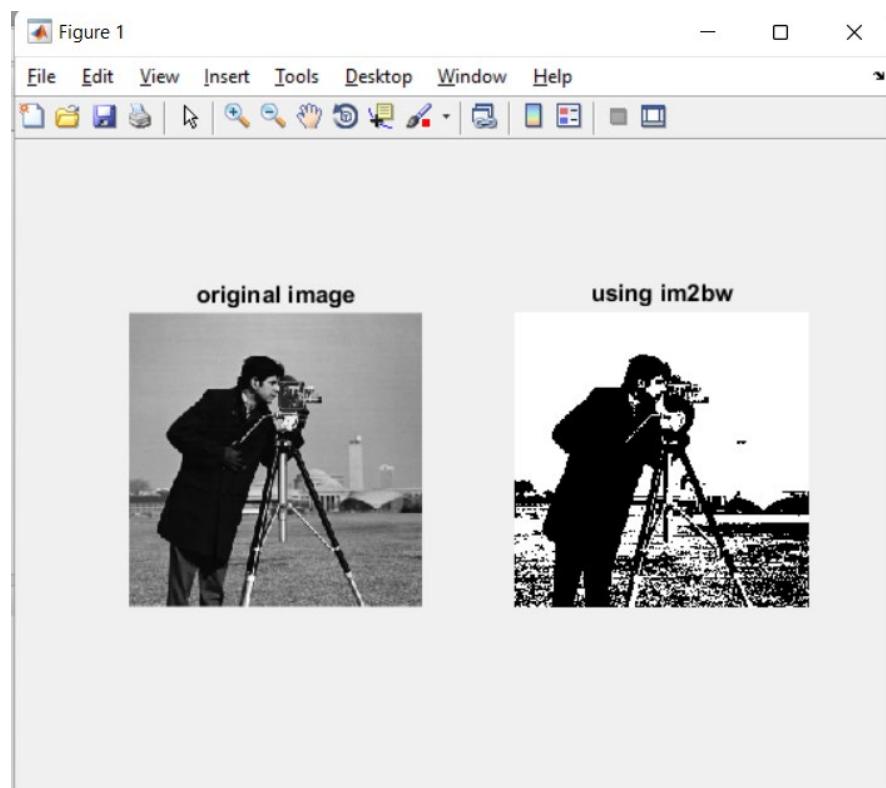
b. Thresholding of an image**Code:**

The screenshot shows the MATLAB environment. The Editor window displays the script file 'thresholding.m' with the following code:

```
Editor - D:\Marwadi\SEM-5\IP-lab\Prg\thresholding.m
negative.m  thresholding.m  +
1 - x=imread('cameraman.tif');
2
3 - subplot(1,2,1);
4 -     imshow (x);
5 -     title('original image');
6
7 - subplot(1,2,2);
8 -     g=im2bw(x);
9 -     imshow (g);
10 -    title('using im2bw');
11 -    fprintf('92000103073 Raj Chhadia');
12
```

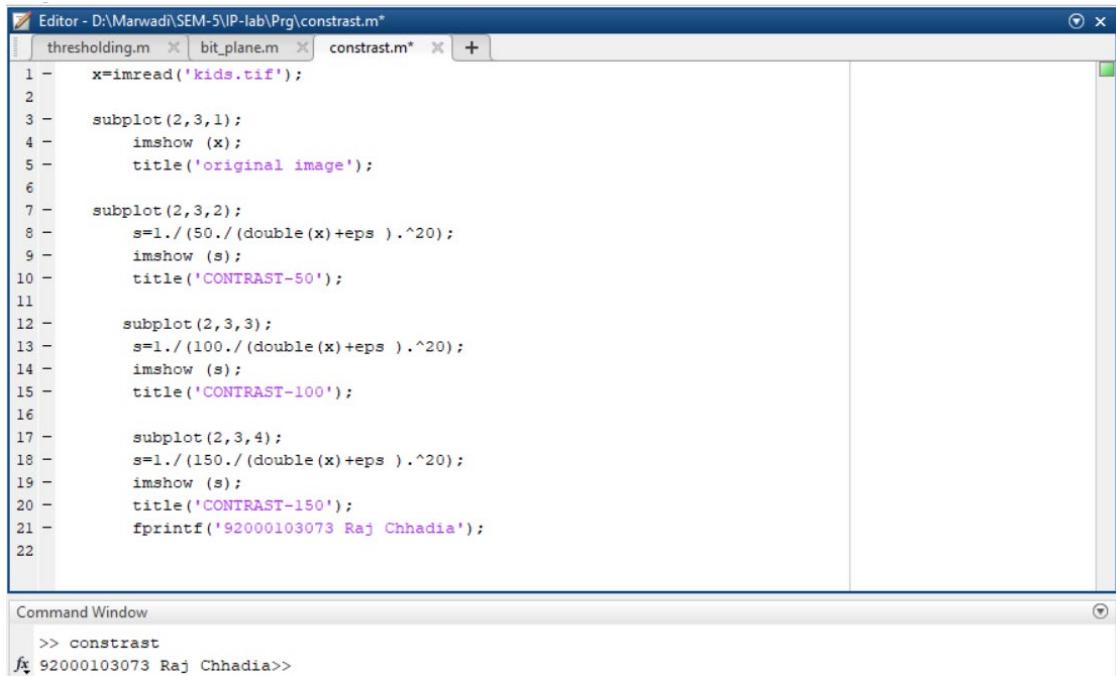
The Command Window shows the execution of the script:

```
Command Window
>> thresholding
fx 92000103073 Raj Chhadia>>
```

Output:

c. Contrast Stretching of an image

Code:



```

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\constrast.m*
thresholding.m  bit_plane.m  constrast.m*  +
1 - x=imread('kids.tif');
2 -
3 - subplot(2,3,1);
4 - imshow (x);
5 - title('original image');
6 -
7 - subplot(2,3,2);
8 - s=1./(50./(double(x)+eps ).^20);
9 - imshow (s);
10 - title('CONTRAST-50');
11 -
12 - subplot(2,3,3);
13 - s=1./(100./(double(x)+eps ).^20);
14 - imshow (s);
15 - title('CONTRAST-100');
16 -
17 - subplot(2,3,4);
18 - s=1./(150./(double(x)+eps ).^20);
19 - imshow (s);
20 - title('CONTRAST-150');
21 - fprintf('92000103073 Raj Chhadia');
22

```

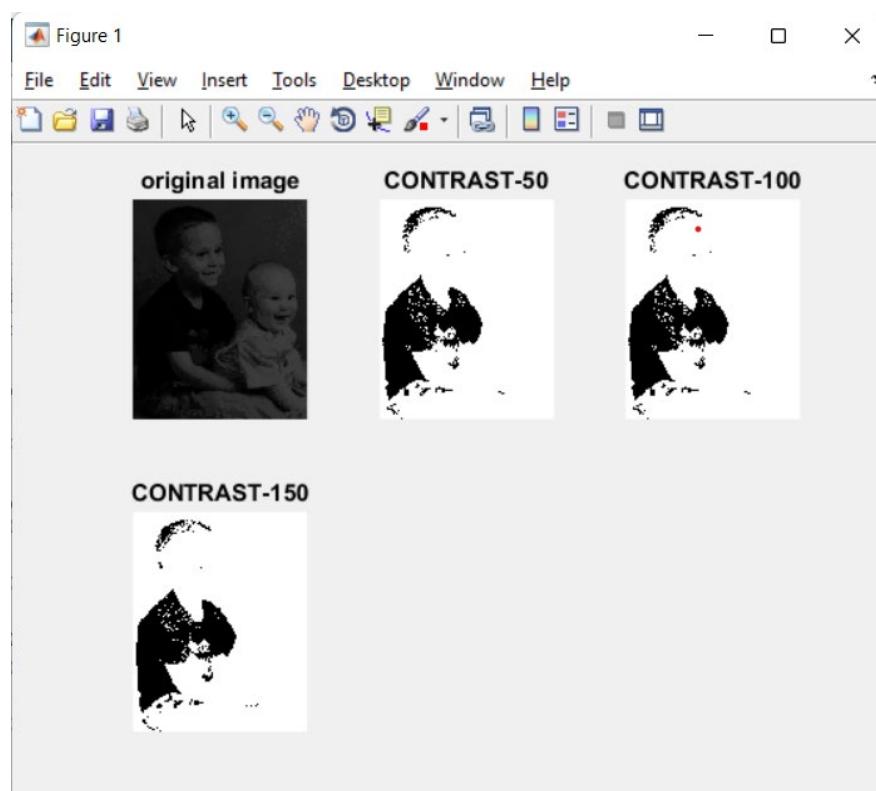
Command Window

```

>> constrast
fx 92000103073 Raj Chhadia>>

```

Output:



Extra:

1. Log Transformation
2. Power-Law Functions
3. Gray-level slicing
4. Bit-plane slicing

1. Log Transformation
Code:

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\log.m

```

log.m + 
1 - x=imread('cameraman.tif');
2
3 - subplot(1,3,1);
4 - imshow (x);
5 - title('original image');
6
7 - subplot(1,3,2);
8 - g=log(1 + double(x));
9 - imshow (g);
10 - title('intermediate');
11
12 - subplot(1,3,3);
13 - gs=im2uint8(mat2gray(g));
14 - imshow (gs);
15 - title('log transformation');
16 - fprintf('92000103073 Raj Chhadia');

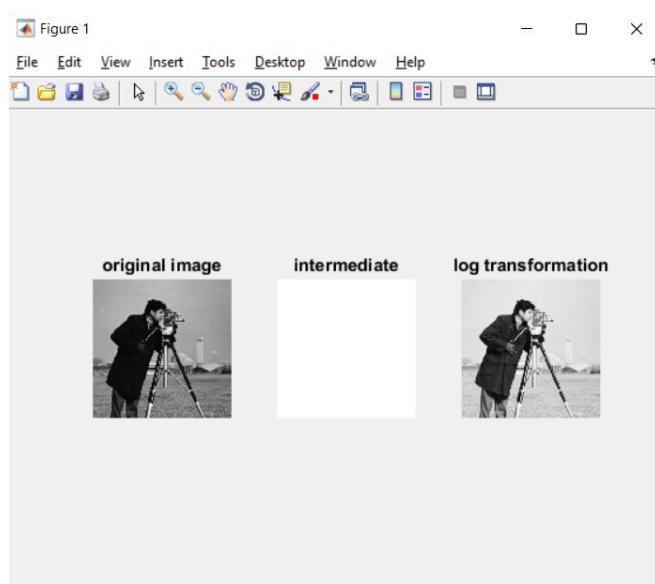
```

Command Window

```

>> log
fx 92000103073 Raj Chhadia>>

```

Output:


2. Power-Law Functions

Code:

Editor - D:\Marwadi\SEM-5\P-lab\Prg\power.m

```

power.m + 
1 - x=imread('cameraman.tif');
2 - subplot(2,3,1);
3 - imshow (x);
4 - title('original image');
5 - subplot(2,3,2);
6 - g=imadjust(x,[0 1],[1,0]);
7 - imshow (g);
8 - title('power trans. r=1');
9 - subplot(2,3,3);
10 - g=imadjust(x,[0.5 0.75],[0 1],0.5);
11 - imshow (g);
12 - title('power transf. r<1');
13 - subplot(2,3,4);
14 - g=imadjust(x,[0.5 0.75],[0.6 1],0.5);
15 - imshow (g);
16 - title('power transf. r>1');
17 - subplot(2,3,5);
18 - g=imadjust(x,[],[],2);
19 - imshow (g);
20 - title('power transf. r>1');
21 - fprintf('92000103073 Raj Chhadia');

```

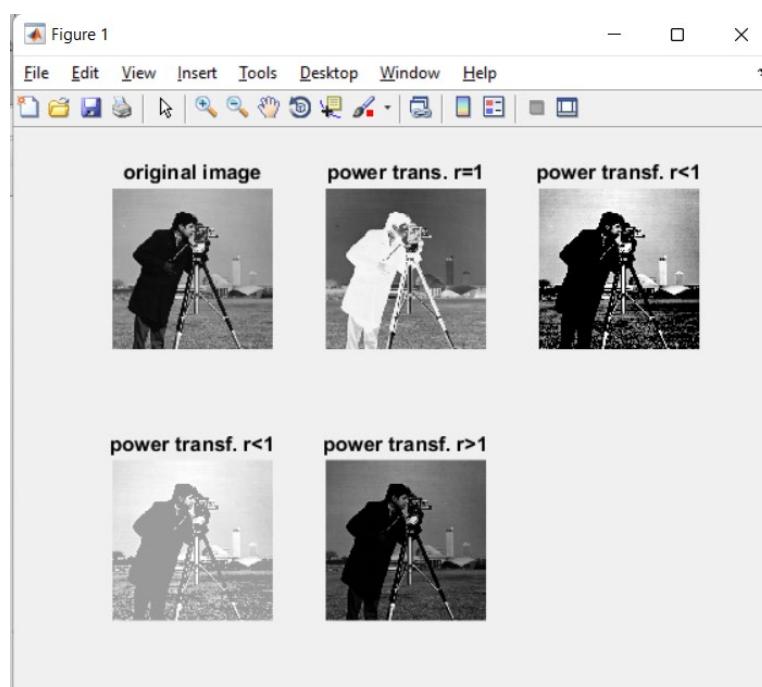
Command Window

```

>> power
fx 92000103073 Raj Chhadia>>

```

Output:



3. Gray-level slicing

Code:

Editor - D:\Marwadi\SEM-5\IP-lab\Prg\gray_lvl.m*

```

1 - r = imread('cameraman.tif');
2 - s = r;
3 - subplot(1,3,1);
4 - imshow(r);
5 - title('original image');
6 - subplot(1,3,2);
7 - for row = 1 : size(r,1)
8 -   for col = 1 : size(r,2)
9 -     if r(row,col) >100 && r(row,col) <200
10 -       s(row,col)=255;
11 -     else
12 -       s(row,col) = 0;
13 -     end
14 -   end
15 - end
16 - imshow(s);
17 - title('Gray level-1');
18 - subplot(1,3,3);
19 - for row = 1 : size(r,1)
20 -   for col = 1 : size(r,2)
21 -     if r(row,col) >100 && r(row,col) <200
22 -       s(row,col)=255;
23 -     else
24 -       s(row,col) = r(row,col);
25 -     end
26 -   end
27 - end
28 - imshow(s);

```

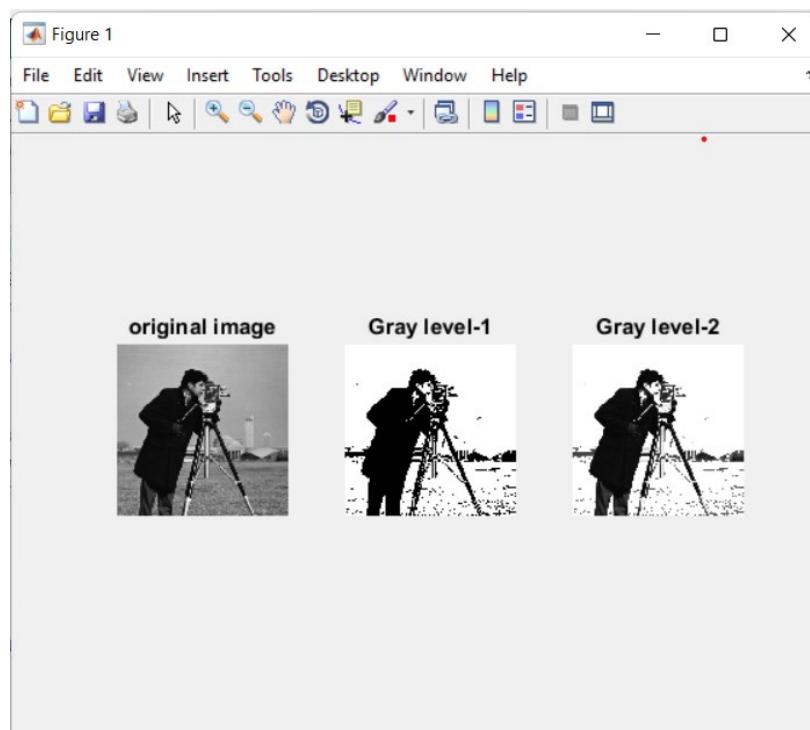
Command Window

```

>> gray_lvl
f 92000103073 Raj Chhadia>>

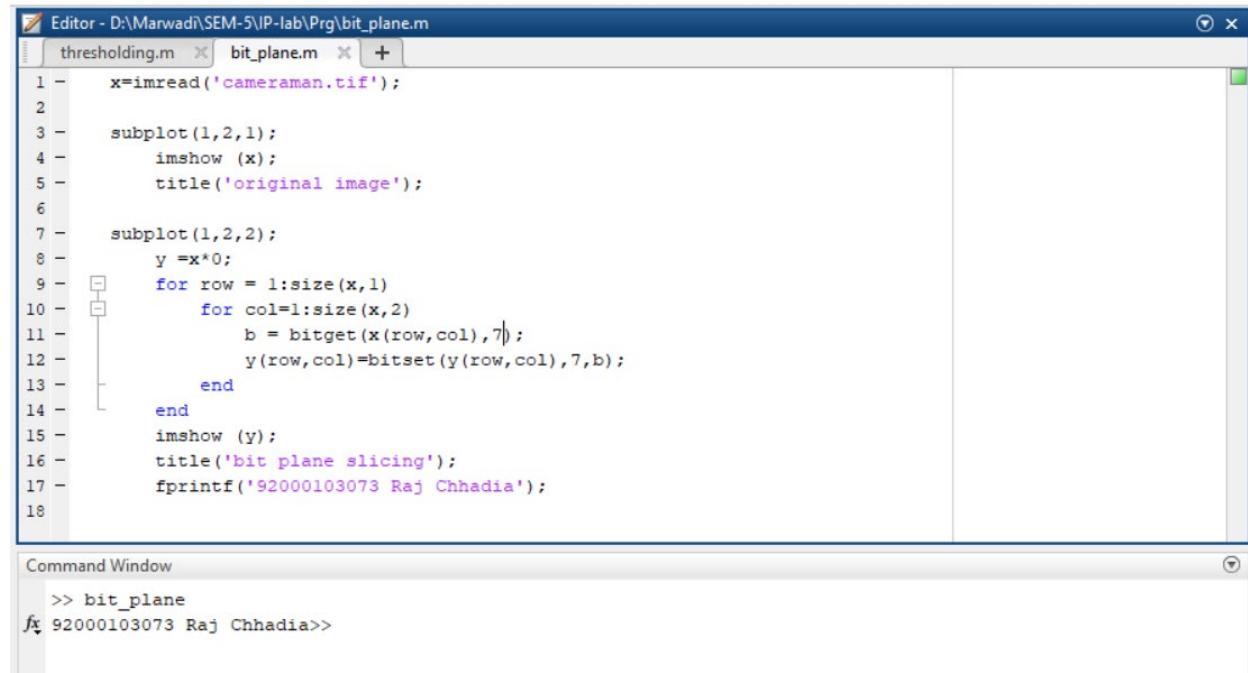
```

Output:



4. Bit-plane slicing

Code:



The screenshot shows the MATLAB environment. The Editor window displays the code for 'bit_plane.m'. The Command Window below shows the execution of the script and its output.

```

Editor - D:\Marwadi\SEM-5\P-lab\Prg\bit_plane.m
thresholding.m  bit_plane.m  +
1 - x=imread('cameraman.tif');
2 -
3 - subplot(1,2,1);
4 - imshow (x);
5 - title('original image');
6 -
7 - subplot(1,2,2);
8 - y =x*0;
9 - for row = 1:size(x,1)
10 -   for col=1:size(x,2)
11 -     b = bitget(x(row,col),7);
12 -     y(row,col)=bitset(y(row,col),7,b);
13 -   end
14 - end
15 - imshow (y);
16 - title('bit plane slicing');
17 - fprintf('92000103073 Raj Chhadia');
18

```

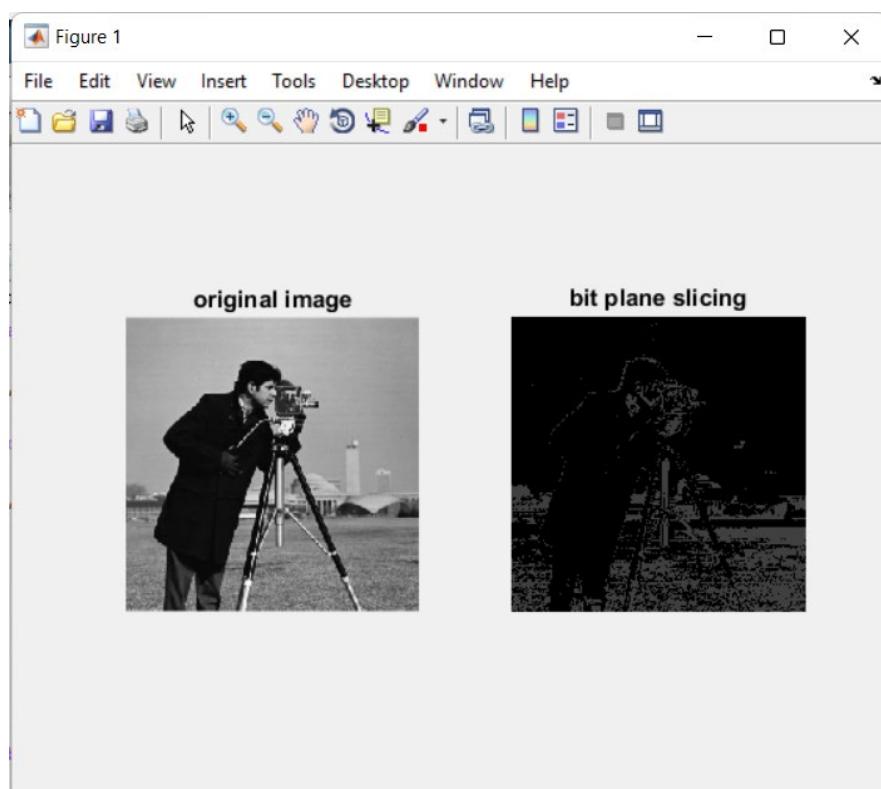
Command Window

```

>> bit_plane
fx 92000103073 Raj Chhadia>>

```

Output:



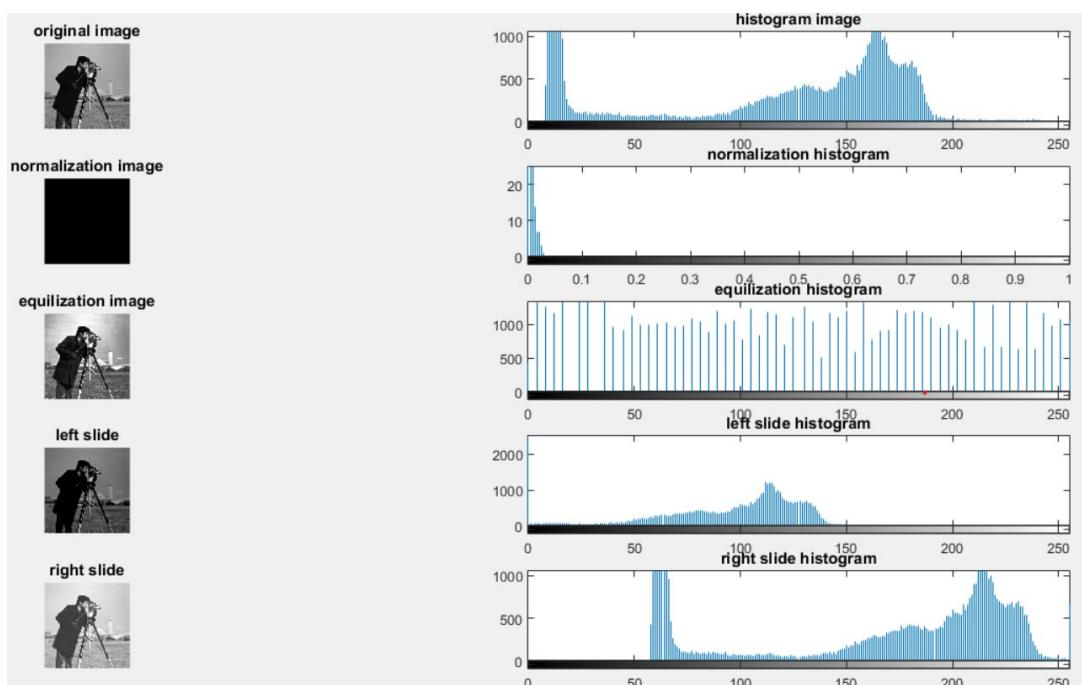
Practical 3

Aim: Write a program for histogram equalization.

Code:

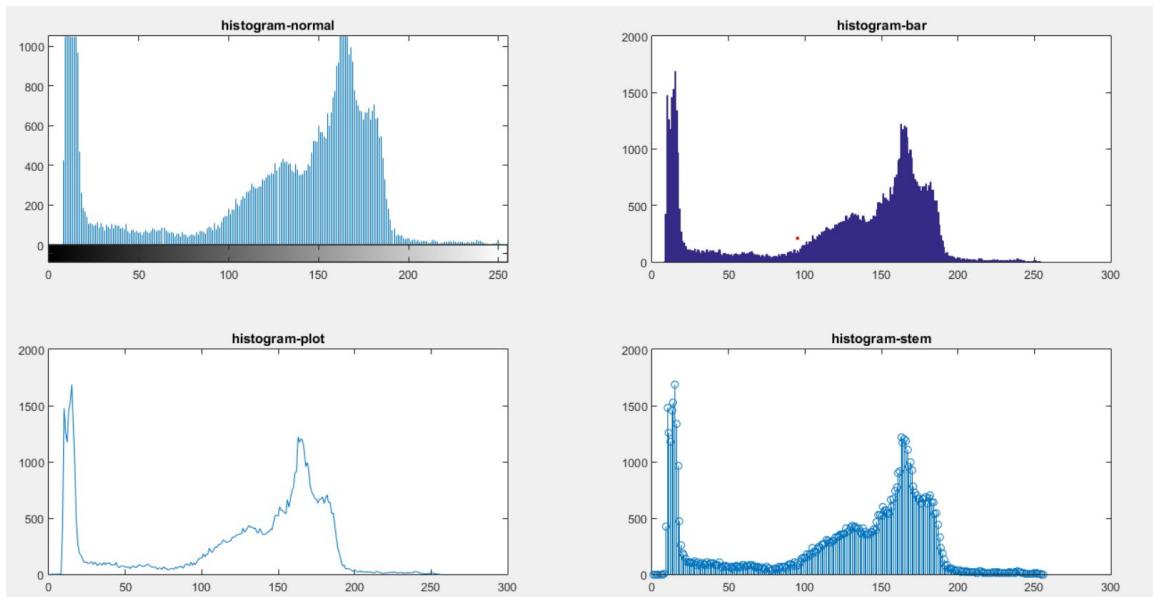
```
histogram_equalization.m
1 r=imread('cameraman.tif');
2 subplot(5,2,1);
3 imshow(r);
4 title('original image');
5
6 subplot(5,2,2);
7 imhist(r);
8 title('histogram image');
9
10 subplot(5,2,3);
11 p=r;
12 total=numel(r);
13 for i=1:size(r,1)
14     for j=1:size(r,2)
15         p(i,j)=r(i,j)/total;
16     end
17 end
18 imshow(p);
19 title('normalization image');
20
21 subplot(5,2,4);
22 p=imhist(r)/numel(r);
23 imhist(p);
24 title('normalization histogram');
25
26 subplot(5,2,5);
27 j=histeq(r);
28 imshow(j);
29 title('equilization image');
30
31 subplot(5,2,6);
32 j=histeq(r);
33 imhist(j);
34 title('equilization histogram');
35
36 subplot(5,2,7);
37 q=r;
38 for i=1:size(r,1)
39     for j=1:size(r,2)
40         q(i,j)=r(i,j)-50;
41     end
42 end
43 imshow(q);
44 title('left slide');
45 subplot(5,2,8);
46 imhist(q);
47 title('left slide histogram');
48 subplot(5,2,9);
49 s=r;
50 for i=1:size(r,1)
51     for j=1:size(r,2)
52         s(i,j)=r(i,j)+50;
53     end
54 end
55 imshow(s);
56 title('right slide');
57 subplot(5,2,10);
58 imhist(s);
59 title('right slide histogram');
60
61 fprintf('92000103073 Raj Chhadia');
```

Output:



Extra:
1. Histogram Types
Code:

```
histogram_types.m
1 - r = imread('cameraman.tif');
2
3 - subplot(2,2,1);
4 - imhist(r);
5 - title('histogram-normal');
6
7 - subplot(2,2,2);
8 - q = bar(p);
9 - title('histogram-bar');
10
11 - subplot(2,2,3);
12 - s = plot(p);
13 - title('histogram-plot');
14
15 - subplot(2,2,4);
16 - t = stem(p);
17 - title('histogram-stem');
```

Output:


2. Histogram Matching

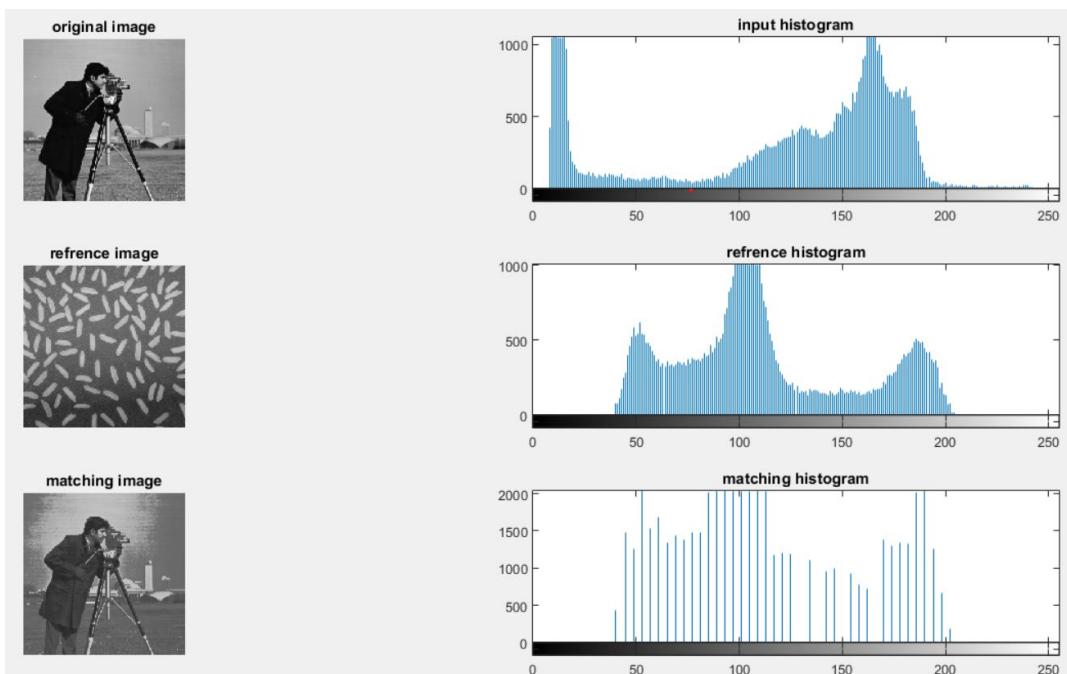
Code:

```

histogram_matching.m  × [+]
1 - r = imread('cameraman.tif');
2 - s = imread('rice.png');
3 -
4 - subplot(3,2,1);
5 - imshow(r);
6 - title('original image');
7 -
8 - subplot(3,2,2);
9 - imhist(r);
10 - title('input histogram');
11 -
12 - subplot(3,2,3);
13 - imshow(s);
14 - title('reference image');
15 -
16 - subplot(3,2,4);
17 - imhist(s);
18 - title('reference histogram');
19 -
20 - subplot(3,2,5);
21 - t = imhistmatch(r,s);
22 - imshow(t);
23 - title('matching image');
24 -
25 - subplot(3,2,6);
26 - imhist(t);
27 - title('matching histogram');
28 - fprintf('92000103073 Raj Chhadia');

```

Output:

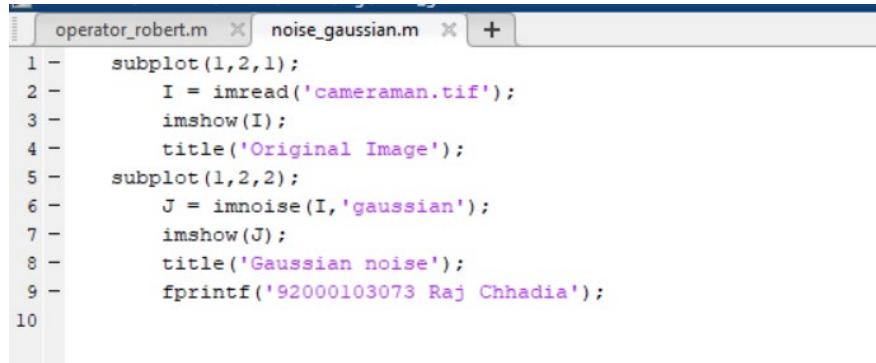


Practical 4

Aim: Write a program to apply various filtering techniques in Matlab.

- **Types of Noise (Gaussian Noise, Poisson noise, Salt & Pepper Noise, Speckle Noise)**

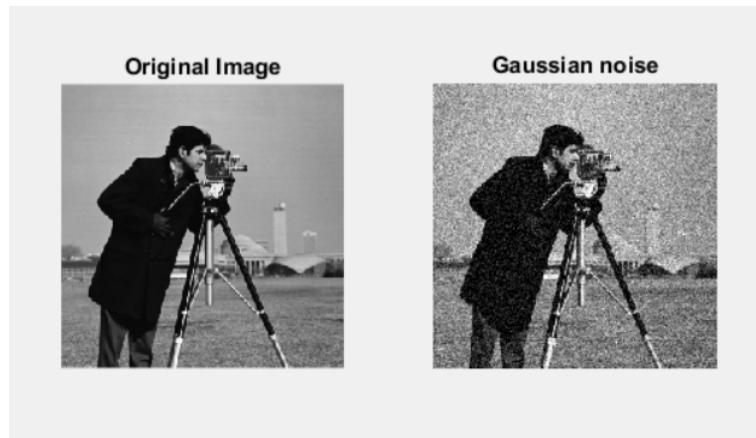
- **Gaussian noise**



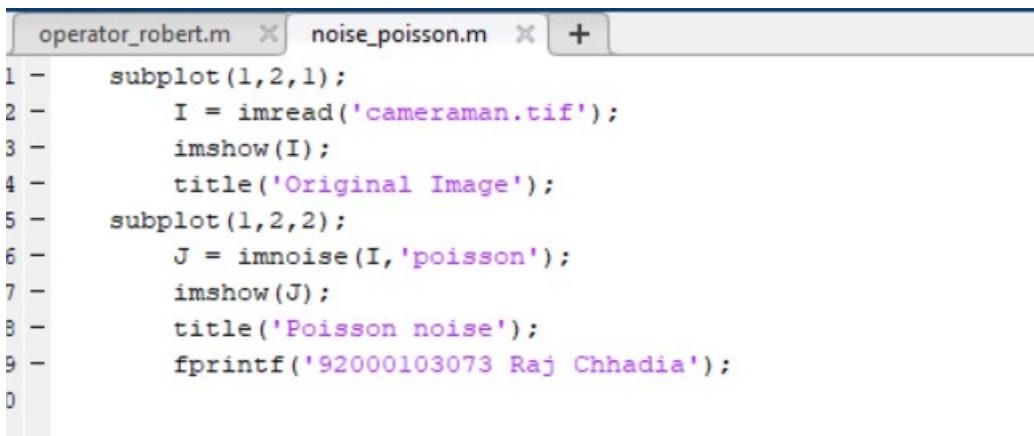
```

operator_robert.m    noise_gaussian.m + 
1 - subplot(1,2,1);
2 -     I = imread('cameraman.tif');
3 -     imshow(I);
4 -     title('Original Image');
5 - subplot(1,2,2);
6 -     J = imnoise(I,'gaussian');
7 -     imshow(J);
8 -     title('Gaussian noise');
9 -     fprintf('92000103073 Raj Chhadia');
10

```



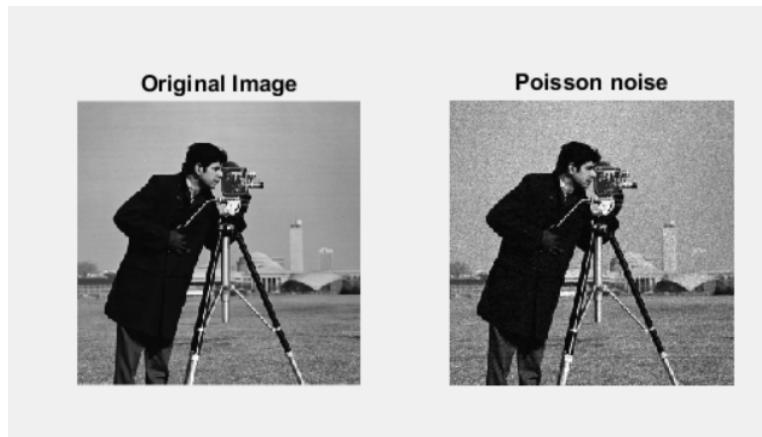
- **Poisson noise**



```

operator_robert.m    noise_poisson.m + 
1 - subplot(1,2,1);
2 -     I = imread('cameraman.tif');
3 -     imshow(I);
4 -     title('Original Image');
5 - subplot(1,2,2);
6 -     J = imnoise(I,'poisson');
7 -     imshow(J);
8 -     title('Poisson noise');
9 -     fprintf('92000103073 Raj Chhadia');
0

```

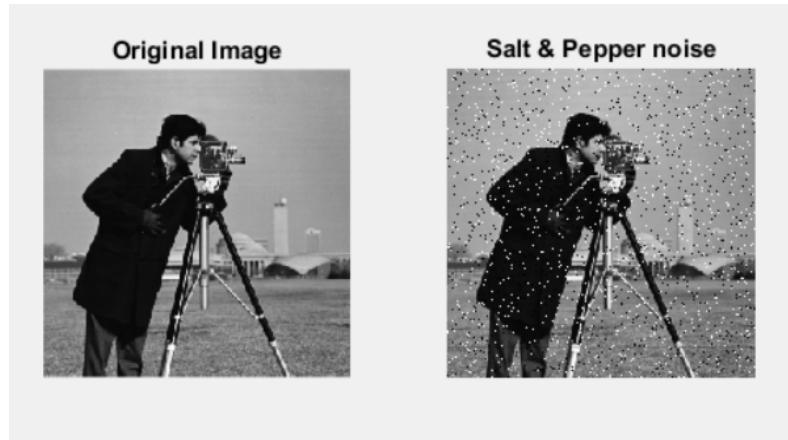


- **Salt & pepper noise**

```

subplot(1,2,1);
I = imread('cameraman.tif');
imshow(I);
title('Original Image');
subplot(1,2,2);
J = imnoise(I,'salt & pepper');
imshow(J);
title('Salt & Pepper noise');
fprintf('92000103073 Raj Chhadia');

```

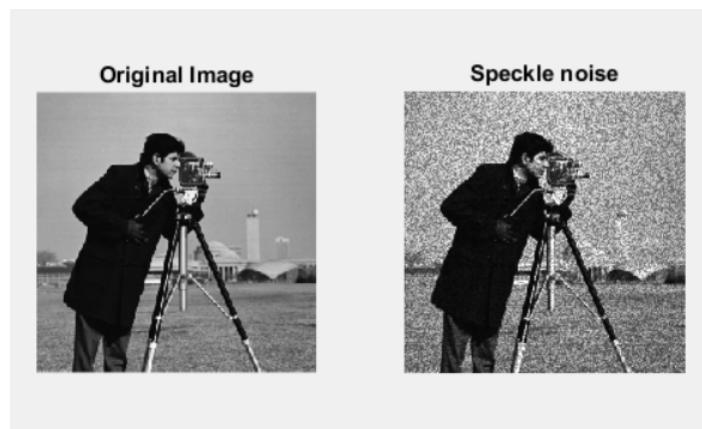


- **Speckle noise**

```

subplot(1,2,1);
I = imread('cameraman.tif');
imshow(I);
title('Original Image');
subplot(1,2,2);
J = imnoise(I,'speckle');
imshow(J);
title('Speckle noise');
fprintf('92000103073 Raj Chhadia');

```



➤ **Spatial Domain - Low Pass Filters / The Smoothing Spatial Filter**

1. Linear Filters / Mean Filter

- Averaging Filter / Standard Average Filter / Arithmetic Mean Filter
 - $N_8(P)$ Neighbor

```
%N8 (P)
I =imread('cameraman.tif');
X = imnoise(I,'salt & pepper');
f=1/9*[1,1,1;1,1,1;1,1,1];

Z=filter2(f,X);
    subplot(1,3,1);
    imshow(I);
    title('Original Image');

    subplot(1,3,2);
    imshow(X);
    title('Noisy Image');

    subplot(1,3,3);
    imshow(uint8(Z));
    title('Denoised Image');
    fprintf('92000103073 Raj Chhadia');
```



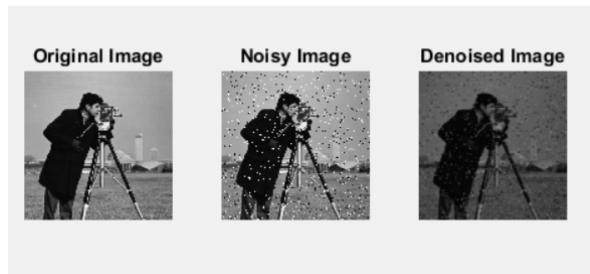
ii. N₄(P) Neighbor

```
%N4 (P)
I =imread('cameraman.tif');
X = imnoise(I,'salt & pepper');
f=1/9*[1,0,1;0,1,0;1,0,1];

Z=filter2(f,X);
 subplot(1,3,1);
 imshow(I);
 title('Original Image');

 subplot(1,3,2);
 imshow(X);
 title('Noisy Image');

 subplot(1,3,3);
 imshow(uint8(Z));
 title('Denoised Image');
 fprintf('92000103073 Raj Chhadia');
```



iii. N_D(P) Neighbor

```
%Nd(P)
I =imread('cameraman.tif');
X = imnoise(I,'salt & pepper');
f=1/9*[0,1,0;1,1,1;0,1,0];

Z=filter2(f,X);
 subplot(1,3,1);
 imshow(I);
 title('Original Image');

 subplot(1,3,2);
 imshow(X);
 title('Noisy Image');

 subplot(1,3,3);
 imshow(uint8(Z));
 title('Denoised Image');
 fprintf('92000103073 Raj Chhadia');
```



b. Weighted Averaging Filter / Gaussian Filter

```
fprintf('92000103073 Raj Chhadia');
I =imread('cameraman.tif');
X =imnoise(I,'gaussian');
f =1/16*[1,2,1;2,4,2;1,2,1];
Z =filter2(f,X);
figure;
subplot(1,3,1);
imshow(I);
title('Original image');

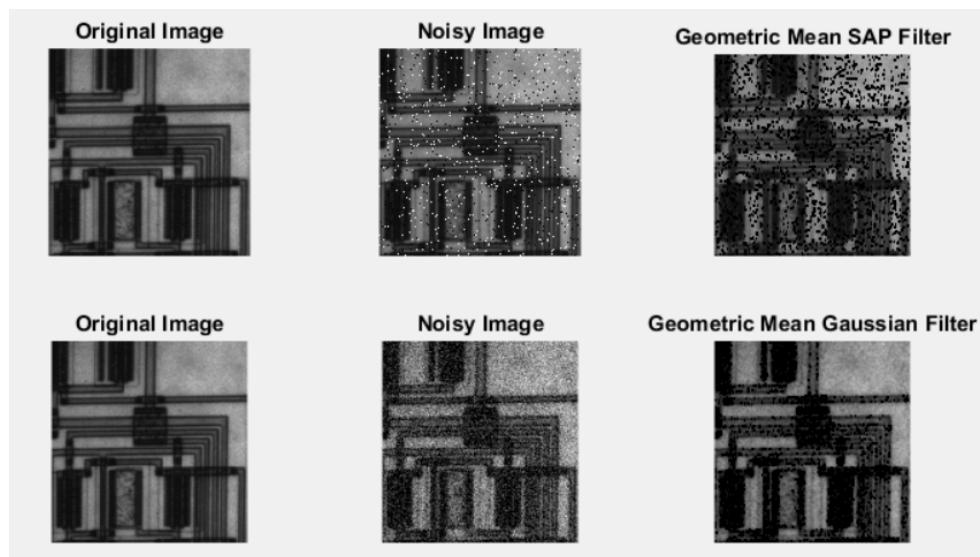
subplot(1,3,2);
imshow(X);
title('Noisy image');

subplot(1,3,3);
imshow(uint8(Z));
title('Denoised image');
```



c. Geometric Mean

```
filter_geometric.m + 
1 fprintf('92000103073 Raj Chhadia');
2 I =imread('circuit.tif');
3 NI = imnoise(I,'salt & pepper');
4 NI1 = im2double(NI);
5 NL = imnoise(I,'gaussian');
6 NL1 = im2double(NL);
7 f=[1,1,1;1,1,1;1,1,1];
8
9 subplot(2,3,1);
10 imshow(I);
11 title('Original Image');
12
13 subplot(2,3,2);
14 imshow(NI);
15 title('Noisy Image');
16
17 subplot(2,3,3);
18 f1= exp(imfilter(log(NI1),f,'replicate')).^(1/9);
19 imshow(f1);
20 title('Geometric Mean SAP Filter');
21
22
23 subplot(2,3,4);
24 imshow(I);
25 title('Original Image');
26
27 subplot(2,3,5);
28 imshow(NL);
29 title('Noisy Image');
30
31 subplot(2,3,6);
32 f2= exp(imfilter(log(NL1),f,'replicate')).^(1/9);
33 imshow(f2);
34 title('Geometric Mean Gaussian Filter');
```



d. Harmonic Mean

```

fprintf('92000103073 Raj Chhadia');
I = imread('circuit.tif');
NI = imnoise(I,'salt & pepper');
NLI = im2double(NI);
NL = imnoise(NI,'gaussian');
NLL = im2double(NL);
f=[1,1,1;1,1,1;1,1,1];

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

subplot(2,3,2);
imshow(NI);
title('Noisy Image');

subplot(2,3,3);

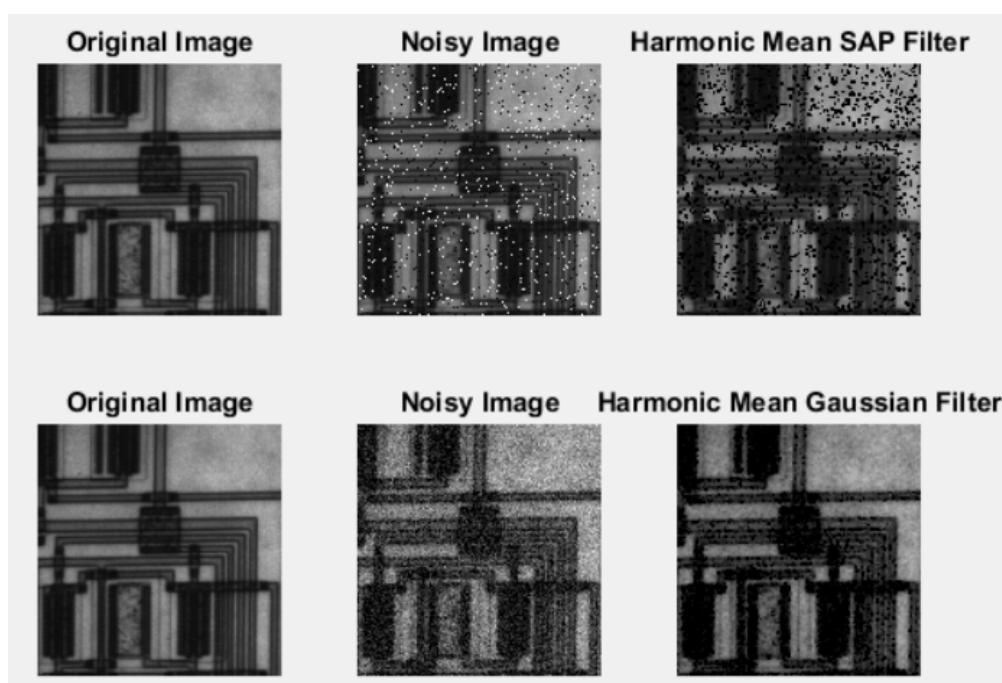
```

```

18 - f1= 3*3./imfilter(1./(NLI+eps),f,'replicate');
19 - imshow(f1);
20 - title('Harmonic Mean SAP Filter');

21
22
23 - subplot(2,3,4);
24 - imshow(I);
25 - title('Original Image');
26
27 - subplot(2,3,5);
28 - imshow(NL);
29 - title('Noisy Image');
30
31 - subplot(2,3,6);
32 - f2= 3*3./imfilter(1./(NLL+eps),f,'replicate');
33 - imshow(f2);
34 - title('Harmonic Mean Gaussian Filter');

```



e. Contraharmonic Mean

```

fprintf('92000103073 Raj Chhadia');
I =imread('circuit.tif');
NI = imnoise(I,'gaussian');
NI1 = im2double(NI);
f=[1,1,1;1,1,1;1,1,1];

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

subplot(3,3,2);
imshow(NI);
title('Noisy Image');

subplot(3,3,3);
c2=imfilter(NI1.^(l+1),f,'replicate');
c3= f2./(imfilter(NI1.^l,f,'replicate')+eps);
imshow(c3);
title('ContraHarmonic Mean Q=1');

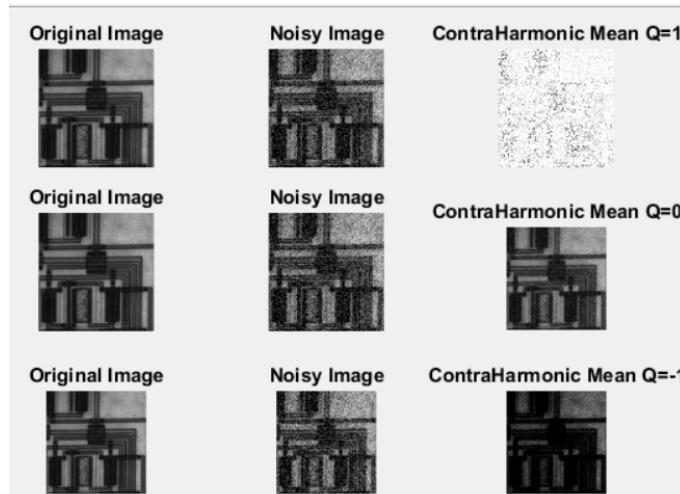
|
subplot(3,3,4);
imshow(I);
title('Original Image');

```

```

24
25 - subplot(3,3,5);
26 - imshow(NI);
27 - title('Noisy Image');
28
29 - subplot(3,3,6);
30 - f2=imfilter(NI1.^(0+l),f,'replicate');
31 - f3= f2./(imfilter(NI1.^0,f,'replicate')+eps);
32 - imshow(f3);
33 - title('ContraHarmonic Mean Q=0');
34
35 - subplot(3,3,7);
36 - imshow(I);
37 - title('Original Image');
38
39 - subplot(3,3,8);
40 - imshow(NI);
41 - title('Noisy Image');
42
43 - subplot(3,3,9);
44 - b2=imfilter(NI1.^(-l+1),f,'replicate');
45 - b3= f2./(imfilter(NI1.^-l,f,'replicate')+eps);
46 - imshow(b3);

```



2. Non-Linear Filters / Order Statistics Filter

a. Median Filtering

```

fprintf('92000103073 Raj Chhadia');
I =imread('cameraman.tif');
NI = imnoise(I,'gaussian');
%NI1 = im2double(NI);
f=[1,1,1;1,1,1;1,1,1];

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

subplot(1,3,2);
imshow(NI);
title('Noisy Image');

subplot(1,3,3);
c2=ordfilt2(NI,5,f);
imshow(c2);
title('Median Filter');

```



b. Max Filtering

```

fprintf('92000103073 Raj Chhadia');
I = imread('cameraman.tif');
NI = imnoise(I,'gaussian');
%NI1 = im2double(NI);
f=[1,1,1;1,1,1;1,1,1];

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

subplot(1,3,2);
imshow(NI);
title('Noisy Image');

subplot(1,3,3);
c2=ordfilt2(NI,9,f);
imshow(c2);
title('Max Filter');

```



c. Min Filtering

```

fprintf('92000103073 Raj Chhadia');
I = imread('cameraman.tif');
NI = imnoise(I,'gaussian');
%NI1 = im2double(NI);
f=[1,1,1;1,1,1;1,1,1];

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

subplot(1,3,2);
imshow(NI);
title('Noisy Image');

subplot(1,3,3);
c2=ordfilt2(NI,1,f);
imshow(c2);
title('Min Filter');

```



d. Mid-point Filtering

```

fprintf('92000103073 Raj Chhadia');
I =imread('cameraman.tif');
NI = imnoise(I,'gaussian');
%NI1 = im2double(NI);
f=[1,1,1;1,1,1;1,1,1];

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

subplot(1,3,2);
imshow(NI);
title('Noisy Image');

subplot(1,3,3);
c2=ordfilt2(NI,1,f);
c3=ordfilt2(NI,9,f);
G=imlincomb(0.5,c2,0.5,c3);
imshow(c2);
title('Midpoint Filter');

```



Practical 8

Aim: Write a program for edge detection.

(Apply Robert Operator, Prewitt Operator, Sobel Operator, Laplacian of Gaussian Operator and Canny Edge Detection methods to perform Edge Detection)

Code:

```

fprintf('92000103073 Raj Chhadia');
% Displaying Input Image
subplot(2,4,1);
input_image = imread('peppers.png');
imshow(input_image);
title('Input Image');

% Convert the RGB image to the grayscale image
subplot(2,4,2);
input_image = rgb2gray(input_image);
imshow(input_image);
title('Input Grayscale Image');

%Apply prewitt Operator
subplot(2,4,3);
filtered_image1 = edge(input_image, 'prewitt');
imshow(filtered_image1);
title('Prewitt Edge Detected Image');

%Apply Sobel Operator
subplot(2,4,4);

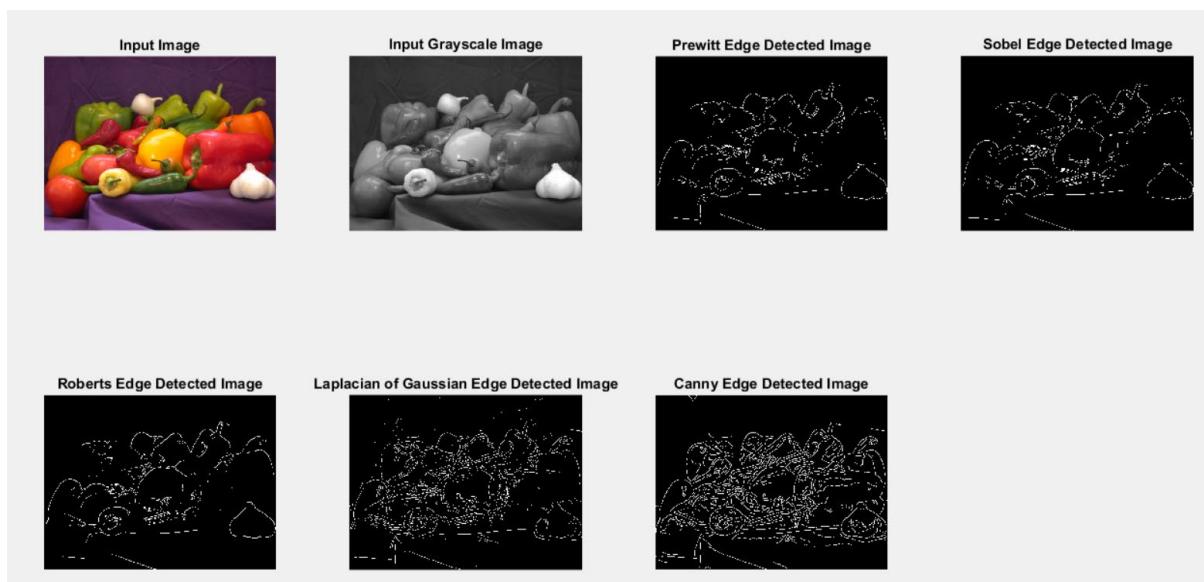
```

```

22 -         filtered_image2 = edge(input_image, 'sobel');
23 -         imshow(filtered_image2);
24 -         title('Sobel Edge Detected Image');
25 -
26 %Apply Roberts Operator
27 -         subplot(2,4,5);
28 -         filtered_image3 = edge(input_image, 'roberts');
29 -         imshow(filtered_image3);
30 -         title('Roberts Edge Detected Image');
31 -
32 %Apply 'Laplacian of Gaussian Operator
33 -         subplot(2,4,6);
34 -         filtered_image4 = edge(input_image, 'log');
35 -         imshow(filtered_image4);
36 -         title('Laplacian of Gaussian Edge Detected Image');
37 -
38 %Apply canny edge Detection algorithm
39 -         subplot(2,4,7);
40 -         filtered_image4 = edge(input_image, 'canny');
41 -         imshow(filtered_image4);
42 -         title('Canny Edge Detected Image');

```

Output:



Practical 9

Aim: Write a program for smoothening and sharpening for 8-bit color image.
 (Apply Laplacian of Gaussian Operator to Perform Edge Detection)

Code:

```

fprintf('92000103073 Raj Chhadia');
subplot(2,3,1);
% Edge detection using Laplacian Filter.
k=imread('peppers.png');
imshow(input_image); title('Input Image');

subplot(2,3,2);
% Convert rgb image to grayscale.
kl=rgb2gray(k);
imshow(kl); title('Input Grayscale Image');

subplot(2,3,3);
% Display the noisy image.
NI =imnoise(kl, 'gaussian');
imshow(NI); title('Noisy Image');

subplot(2,3,4)
17 -   subplot(2,3,4)
18 -     % Create the gaussian Filter.
19 -     GI=fspecial('gaussian', 5, 1);
20 -     % Define the Laplacian Filter.
21 -     Lap=[0 -1 0; -1 4 -1; 0 -1 0];
22 -     % Convolve the noisy image
23 -     % with Gaussian Filter first.
24 -     Output1=conv2(NI, GI, 'same');
25 -     % Display the Gaussian of noisy _ image.
26 -     imshow(uint8(Output1)); title('Apply Gaussian Filter');

27 -
28 -   subplot(2,3,5)
29 -     % Convolve the resultant
30 -     % image with Laplacian filter.
31 -     Output2=conv2(Output1, Lap, 'same');
32 -     % Display the Laplacian of Gaussian resultant image.
33 -     imshow(Output2); title('Apply Laplacian Operator');

```

Output:

