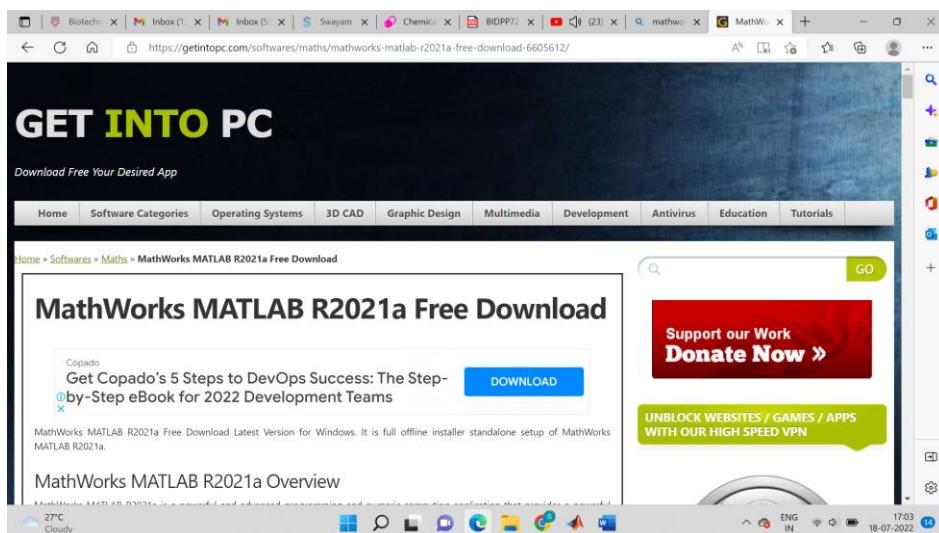


BID-701 MATLAB in Bioinformatics

Assignment 1: Installation of MATLAB

Step 1: Search for 2021 link of MATLAB application on google and download the zip file from the link:

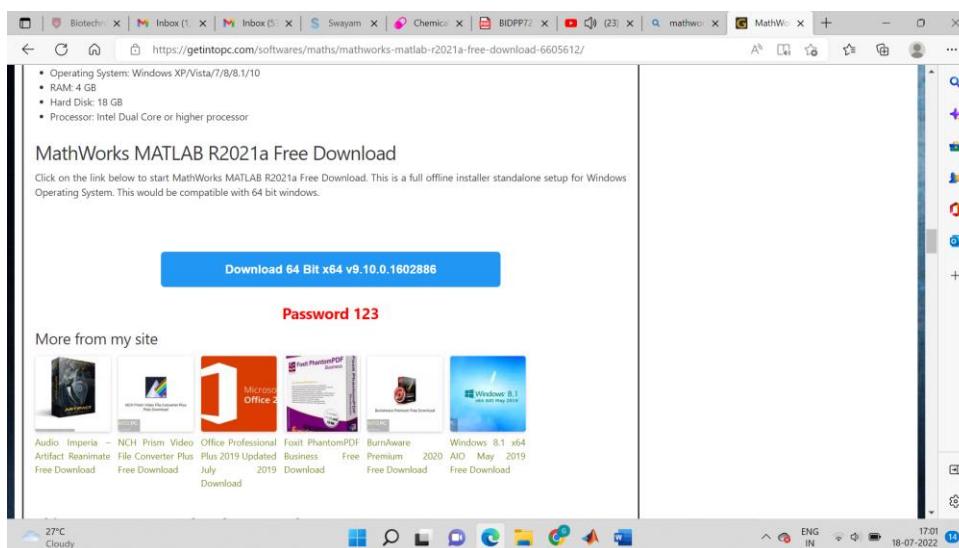
https://51-68-135-75.xyz/Getintopc.com/MathWorks_MATLAB_R2021a_v9.10.0.1602886.rar?md5=GSTMn7u2otVTo4dgtfNEAw&expires=1660113751



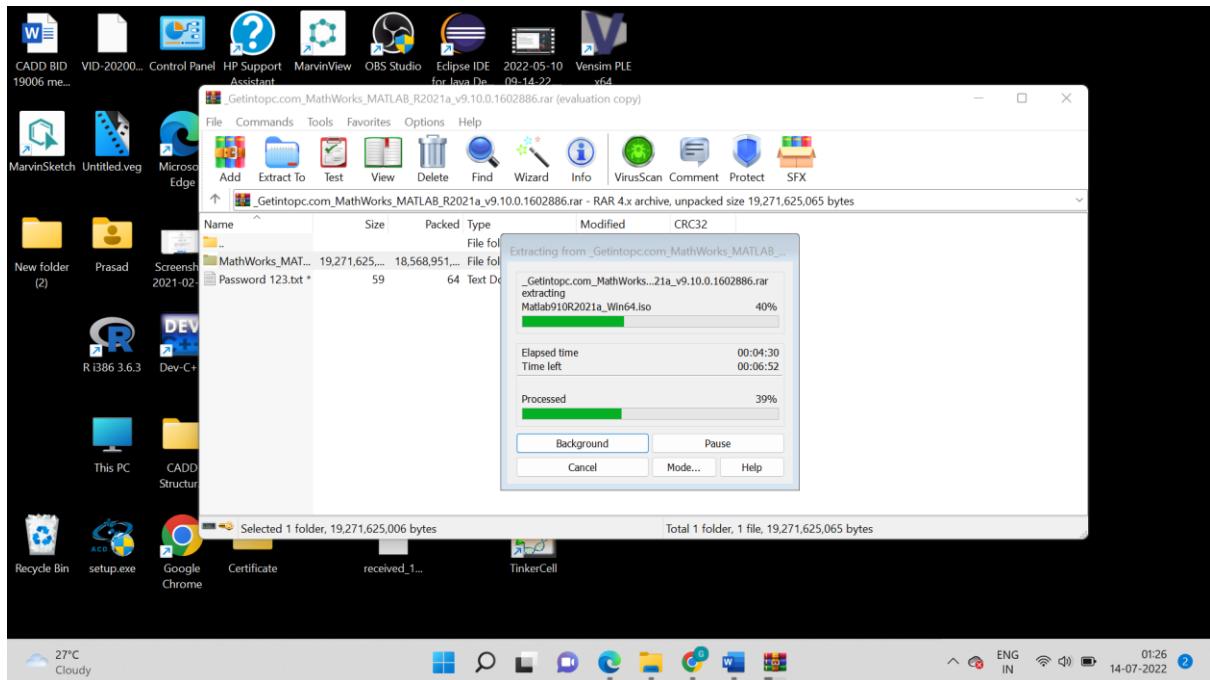
Check your system requirement:

System Requirements for MathWorks MATLAB R2021a

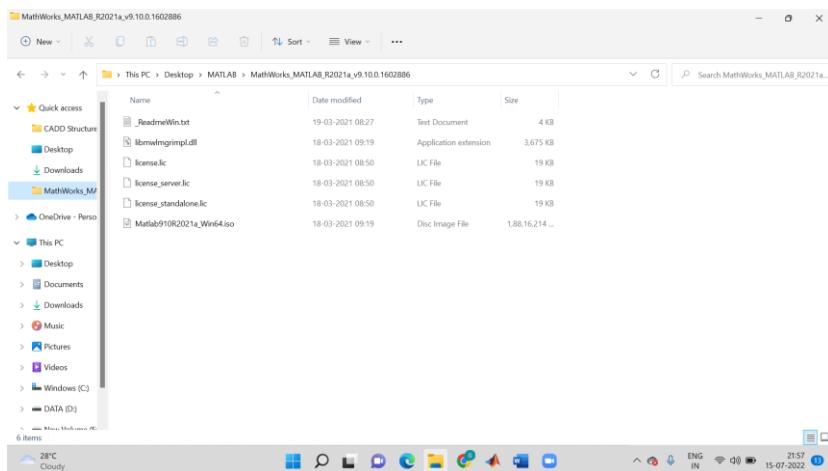
- Operating System: Windows XP/Vista/7/8/8.1/10
- RAM: 4 GB
- Hard Disk: 18 GB
- Processor: Intel Dual Core or higher processor



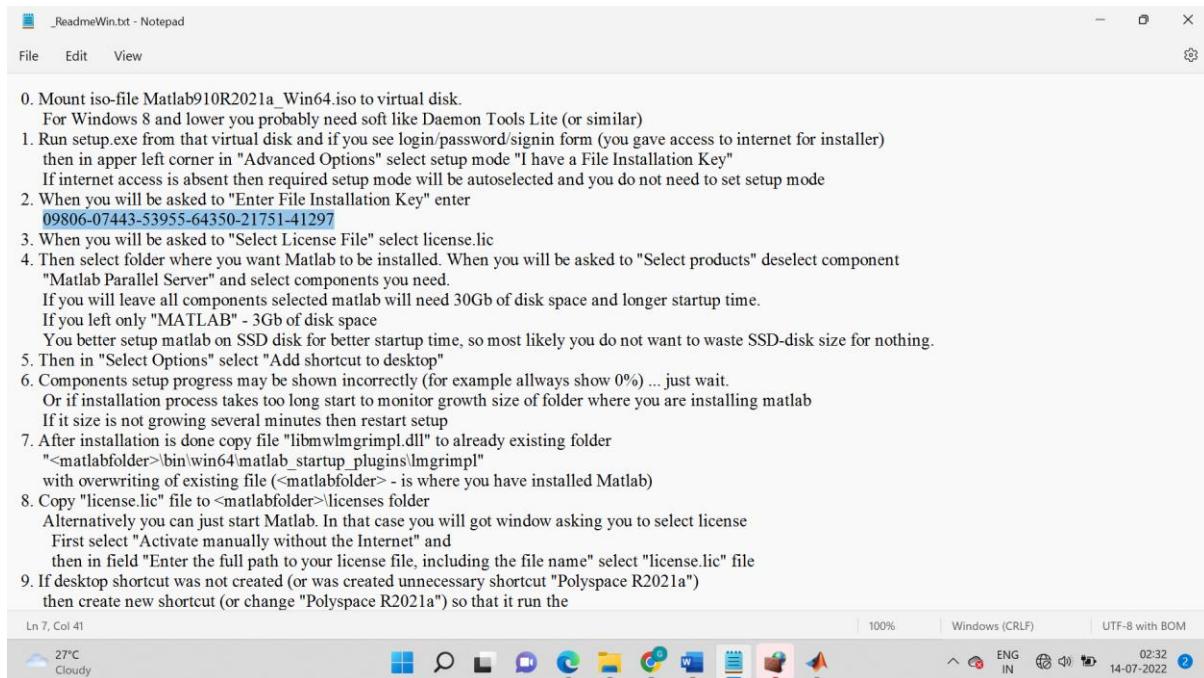
Step 2: Make a folder named MATLAB on desktop and extract all the zip files from the link to the folder on desktop:



Step 3: Now open the ReadmeWin text file from the desktop MATLAB folder:

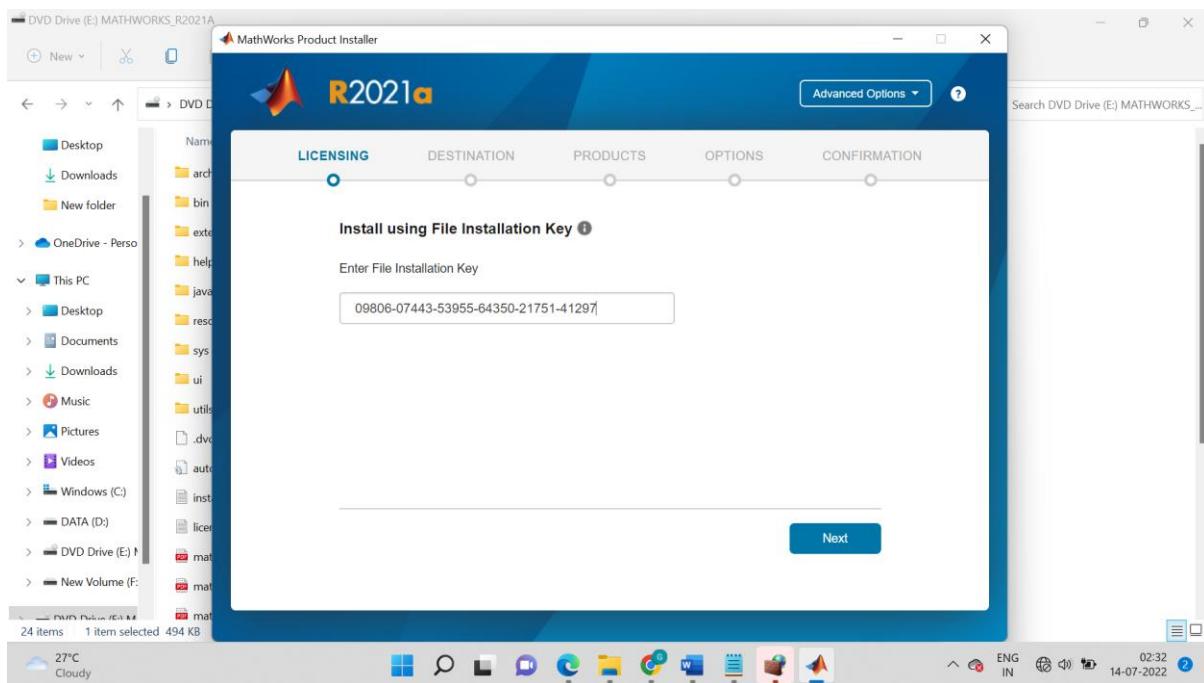


Step 4: Now copy the installation key from the text file:

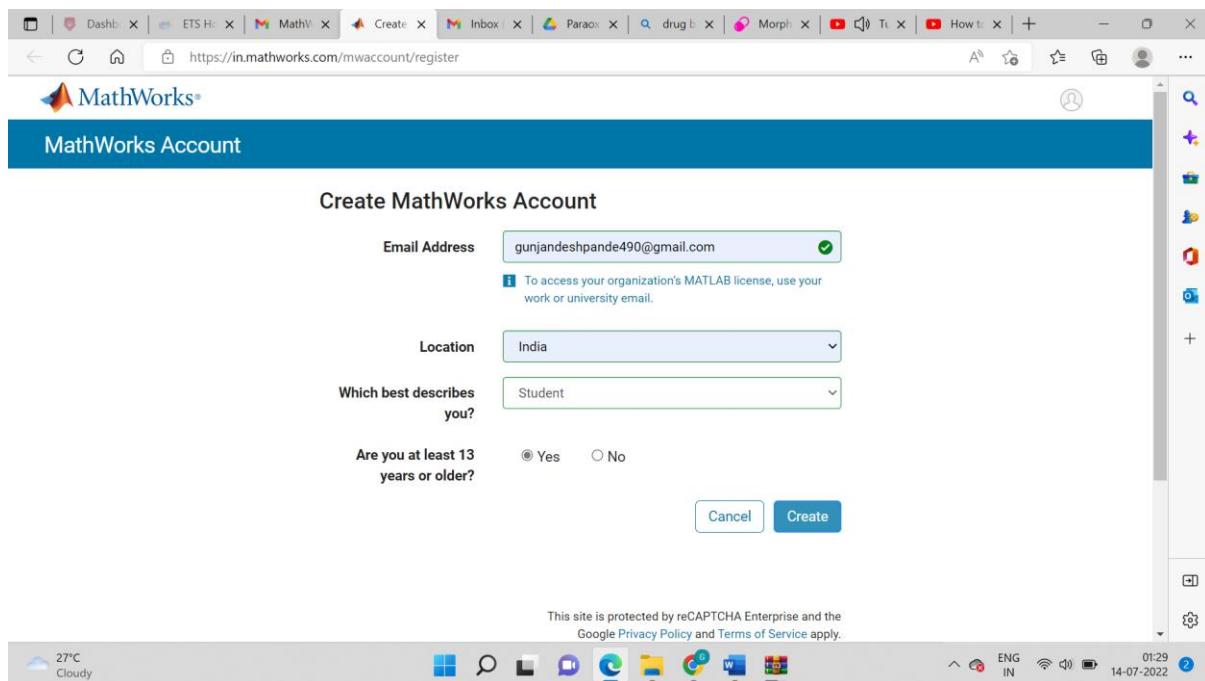


```
0. Mount iso-file Matlab910R2021a_Win64.iso to virtual disk.  
For Windows 8 and lower you probably need soft like Daemon Tools Lite (or similar)  
1. Run setup.exe from that virtual disk and if you see login/password/signin form (you gave access to internet for installer) then in upper left corner in "Advanced Options" select setup mode "I have a File Installation Key"  
If internet access is absent then required setup mode will be autoselected and you do not need to set setup mode  
2. When you will be asked to "Enter File Installation Key" enter  
09806-07443-53955-64350-21751-41297  
3. When you will be asked to "Select License File" select license.lic  
4. Then select folder where you want Matlab to be installed. When you will be asked to "Select products" deselect component "Matlab Parallel Server" and select components you need.  
If you will leave all components selected matlab will need 30Gb of disk space and longer startup time.  
If you left only "MATLAB" - 3Gb of disk space  
You better setup matlab on SSD disk for better startup time, so most likely you do not want to waste SSD-disk size for nothing.  
5. Then in "Select Options" select "Add shortcut to desktop"  
6. Components setup progress may be shown incorrectly (for example always show 0%) ... just wait.  
Or if installation process takes too long start to monitor growth size of folder where you are installing matlab  
If it size is not growing several minutes then restart setup  
7. After installation is done copy file "libmwlmgrimpl.dll" to already existing folder  
"<matlabfolder>/bin/win64/matlab_startup_plugins/lmgrimpl"  
with overwriting of existing file (<matlabfolder> - is where you have installed Matlab)  
8. Copy "license.lic" file to <matlabfolder>/licenses folder  
Alternatively you can just start Matlab. In that case you will get window asking you to select license  
First select "Activate manually without the Internet" and  
then in field "Enter the full path to your license file, including the file name" select "license.lic" file  
9. If desktop shortcut was not created (or was created unnecessary shortcut "Polyspace R2021a")  
then create new shortcut (or change "Polyspace R2021a") so that it run the
```

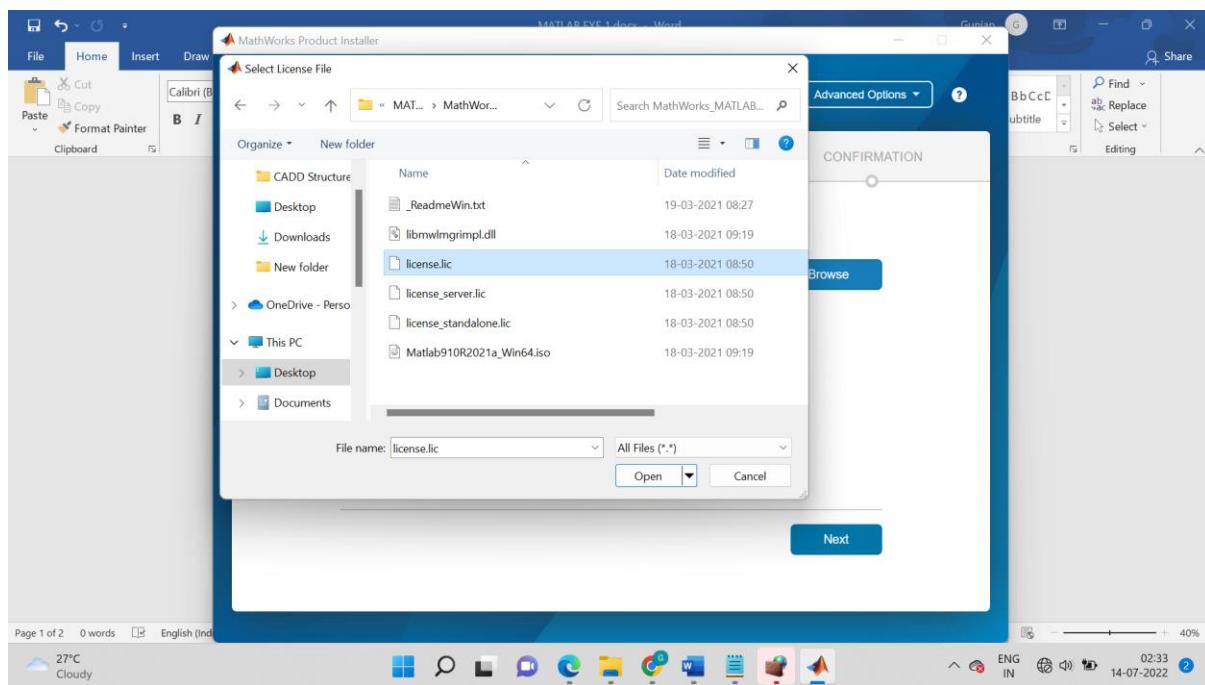
Step 5: You have to turn off your internet and open the MathWorks Product Installer and paste the installation key:



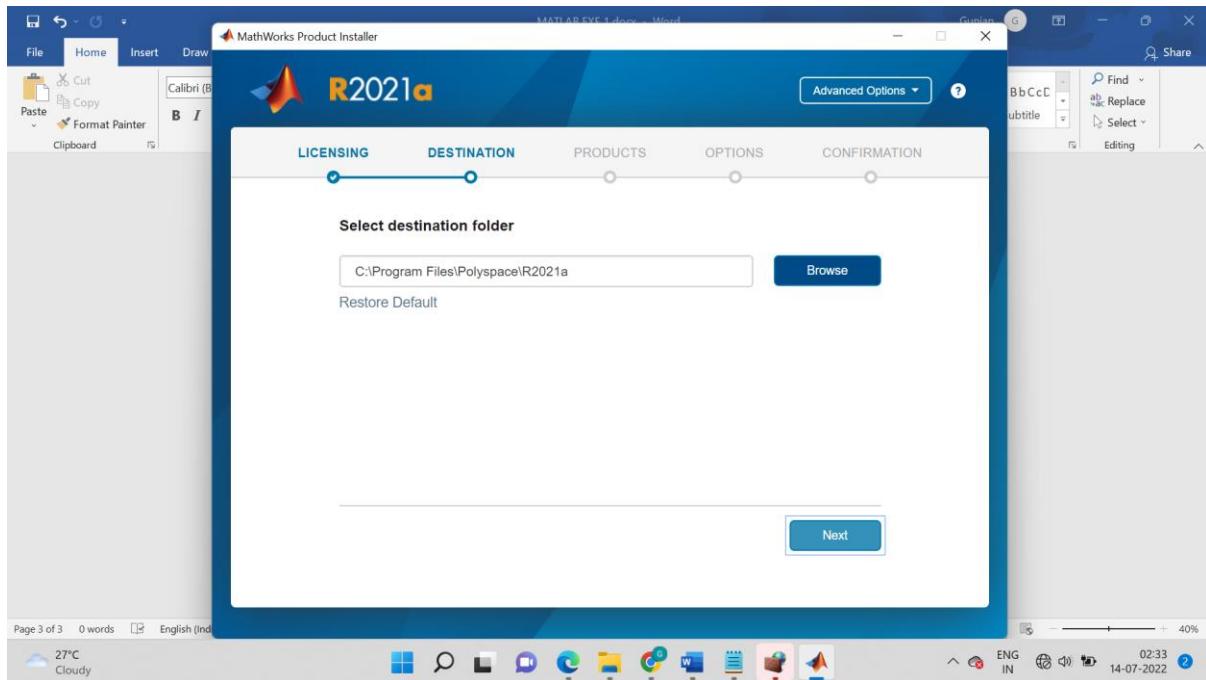
Step 6: Now create a Mathworks account :



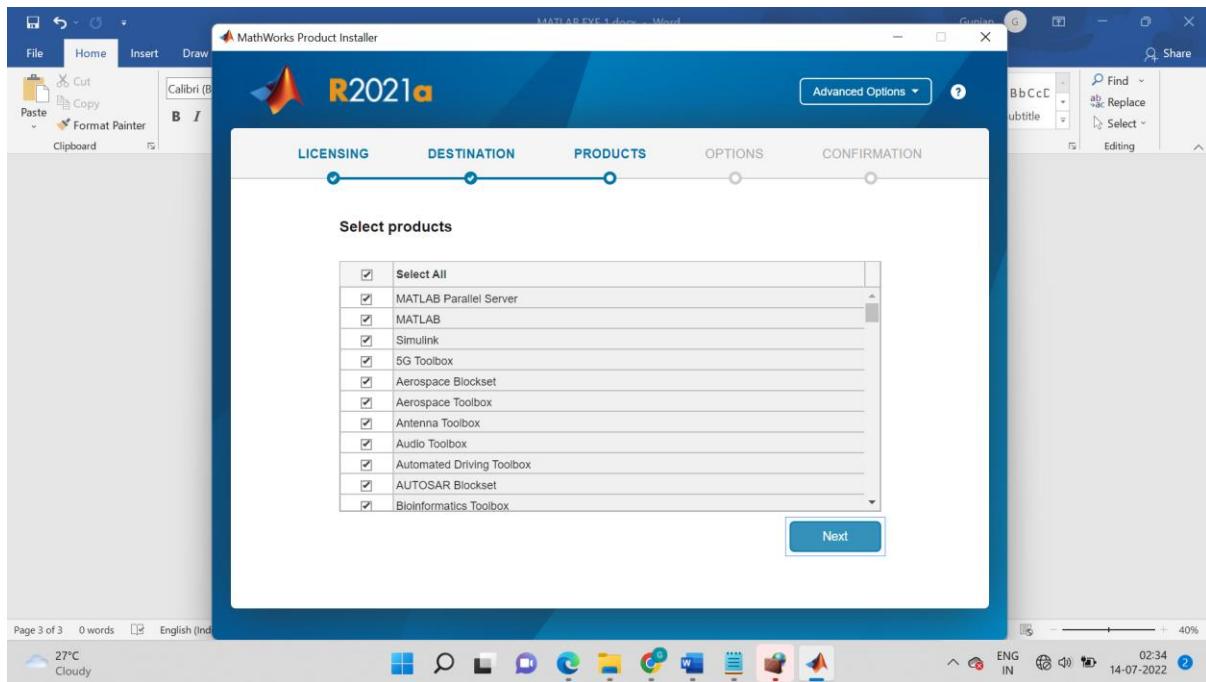
Step 7: Now go add the license file in the account :



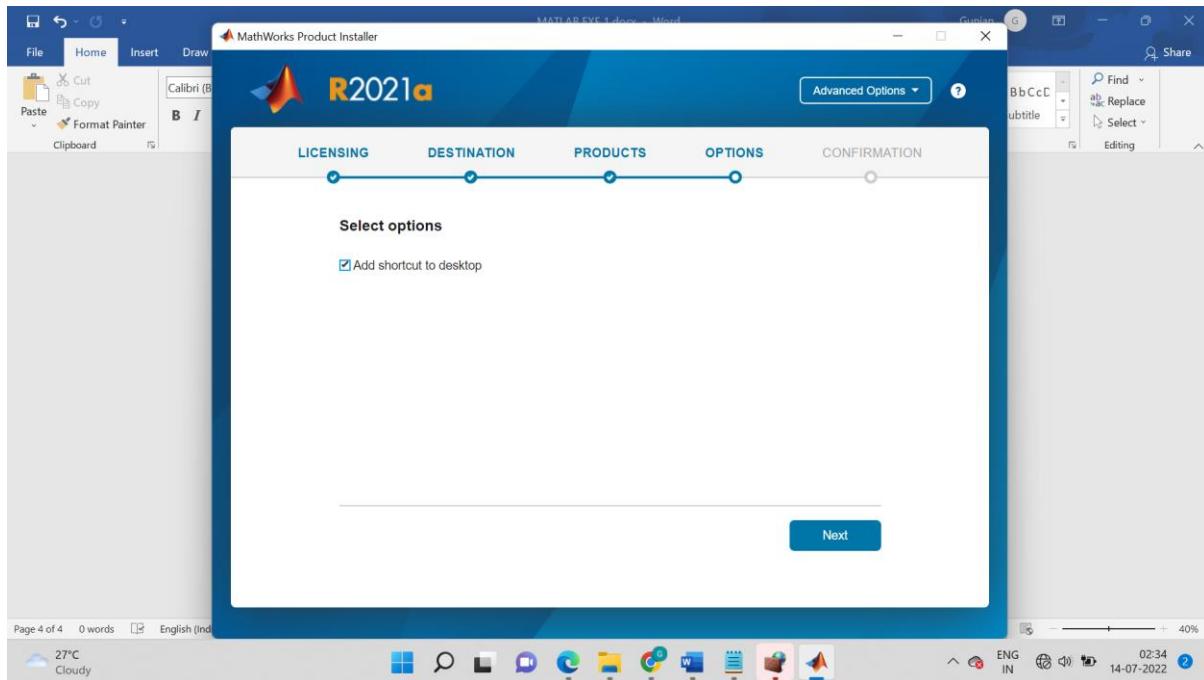
Step 8: Copy paste your R2021a file destination:



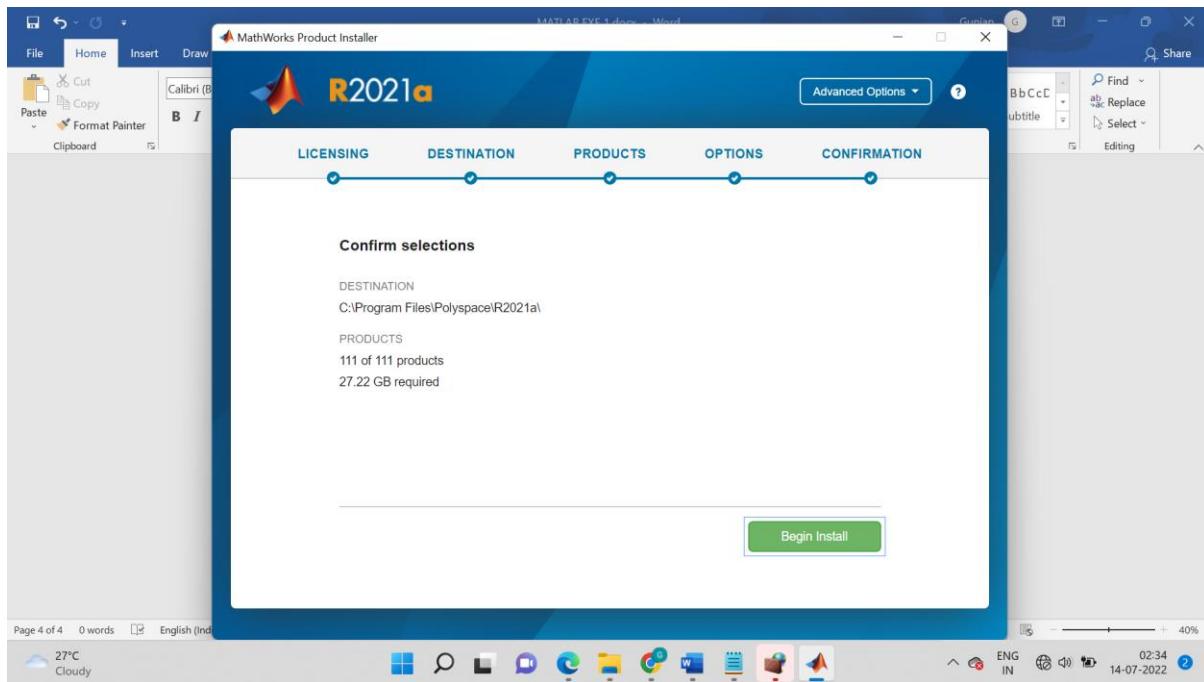
Step 9: Now select all the products so that it gets downloaded:



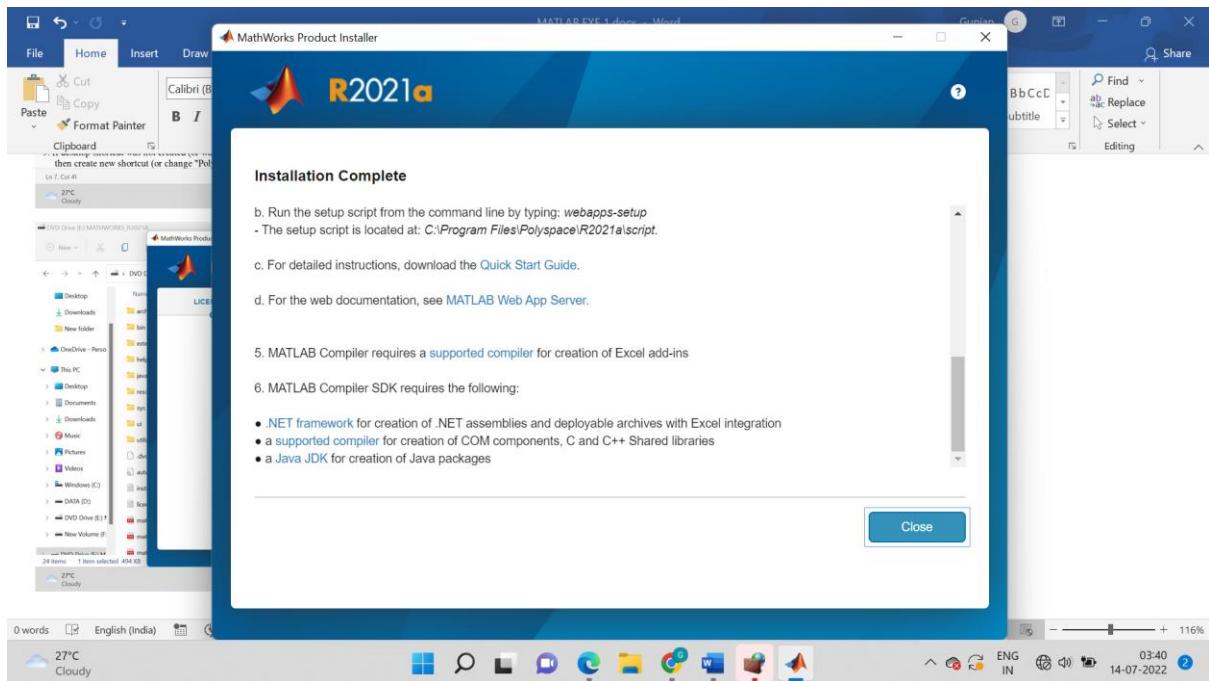
Step 10: Tick the box so that MATLAB gets added to the desktop:



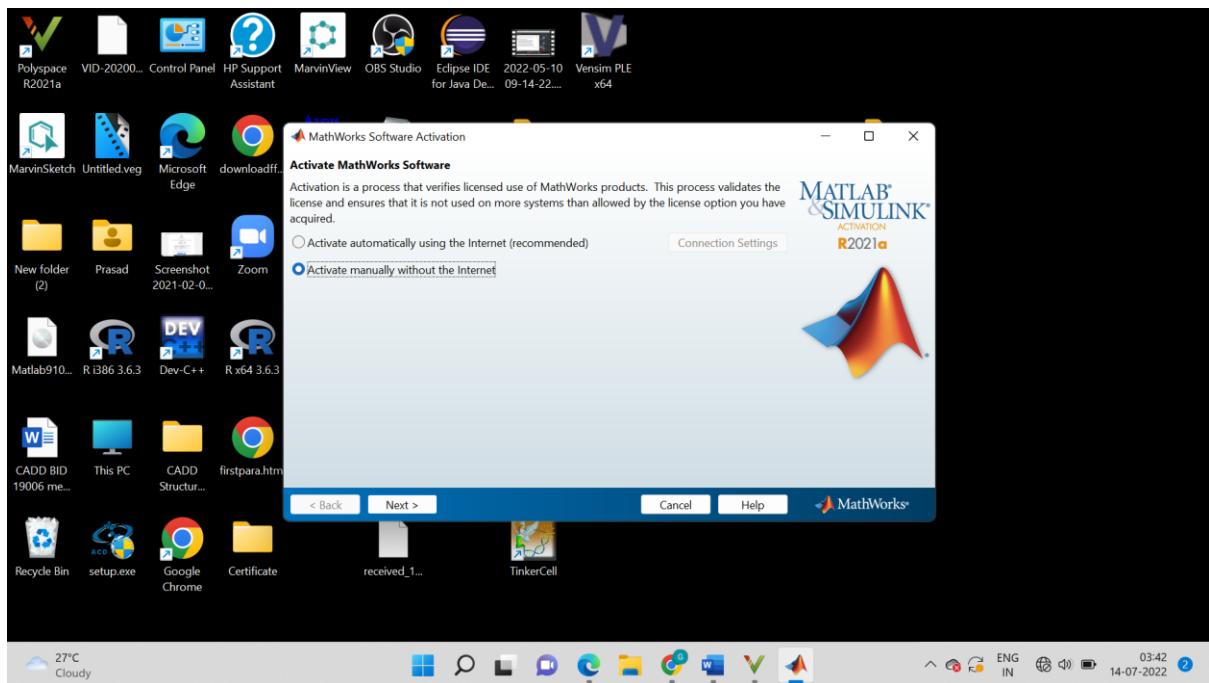
Step 11: Click on begin installation:

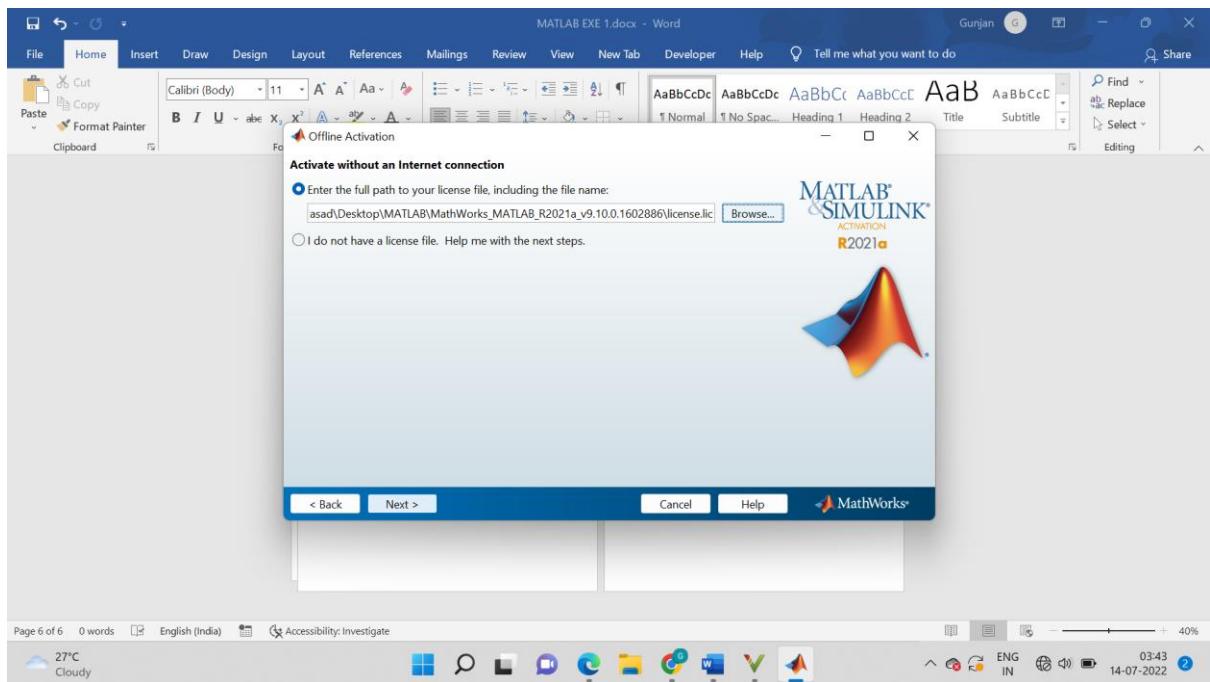


Step 12: Now the installation is completed so click on close:

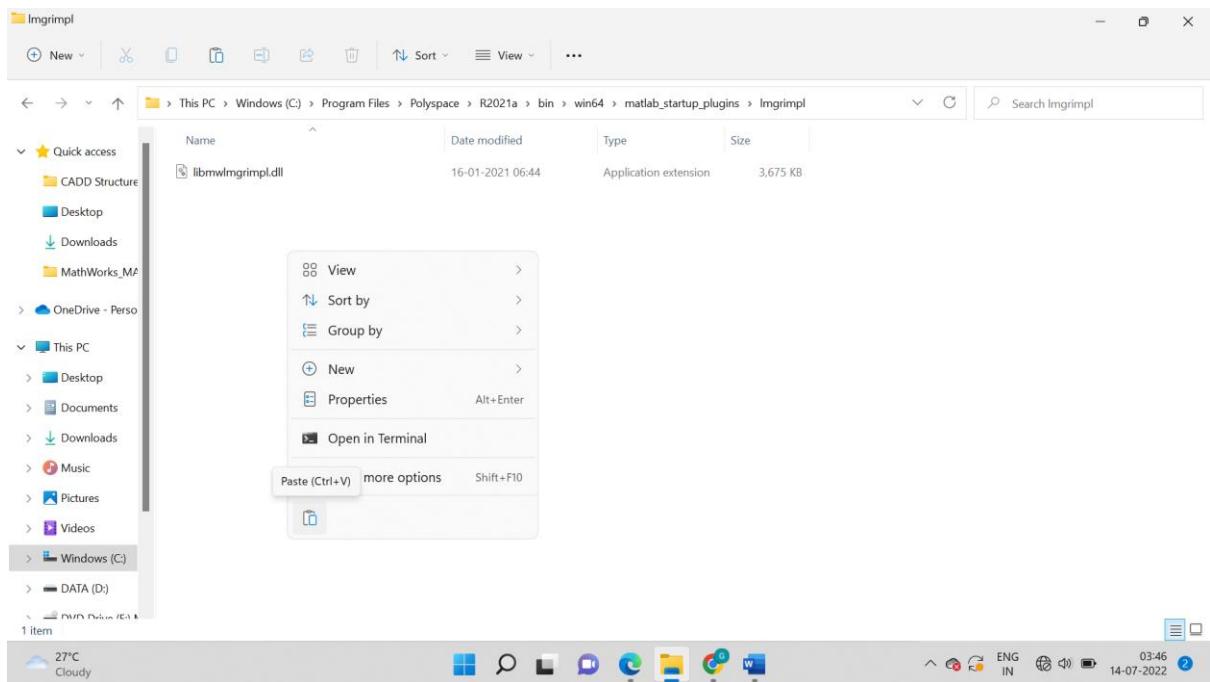


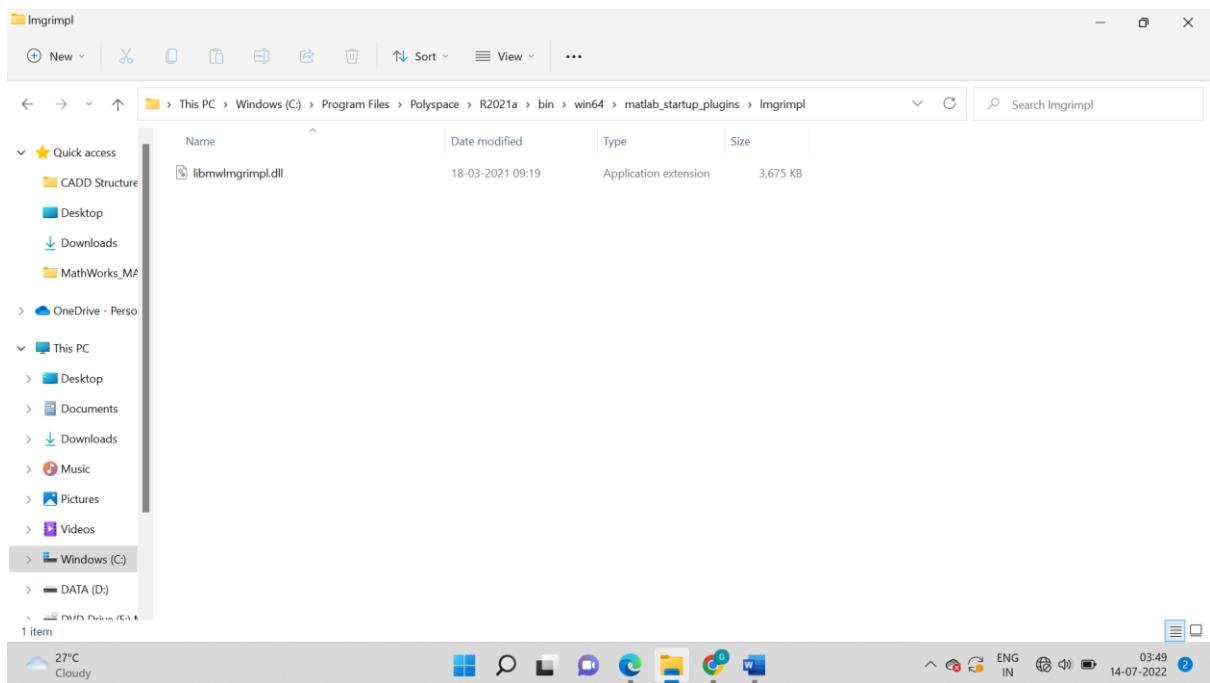
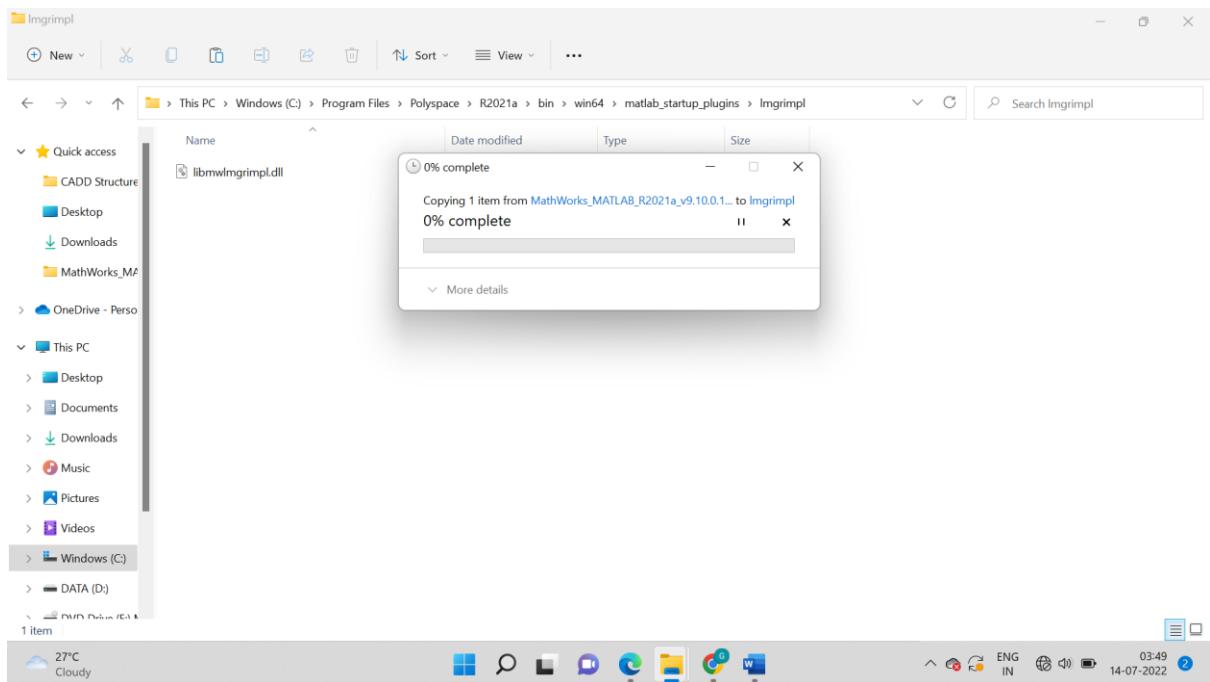
Step 13: Click on activate manually without the internet:

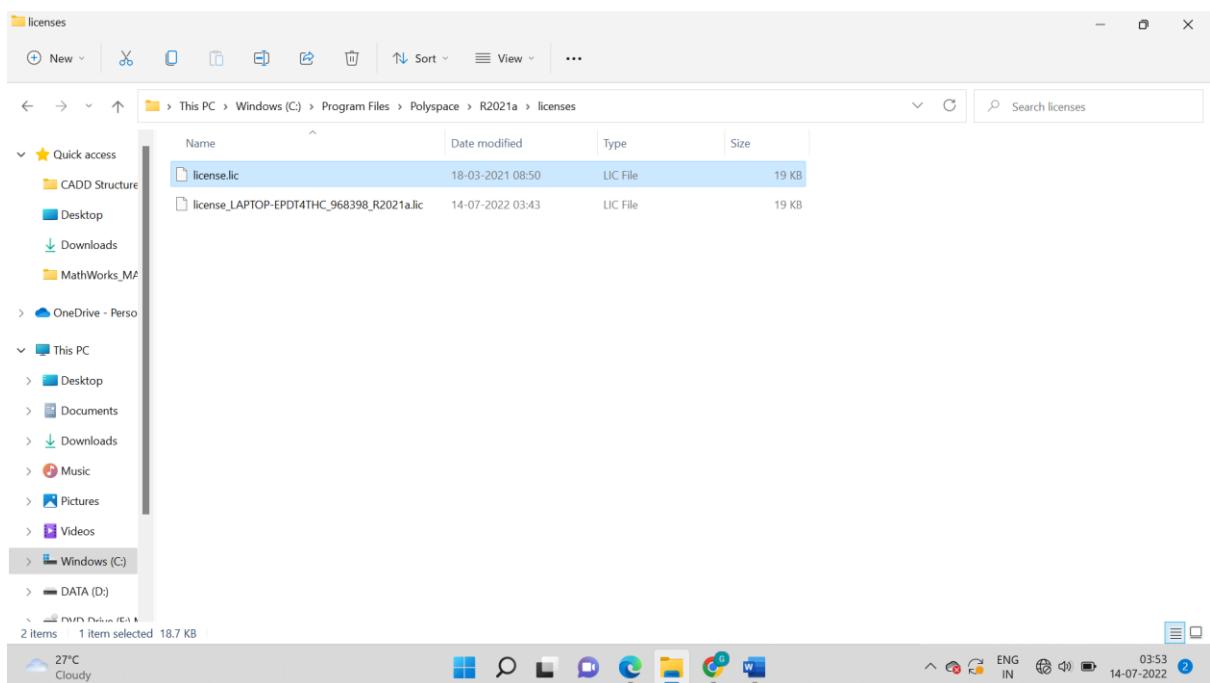
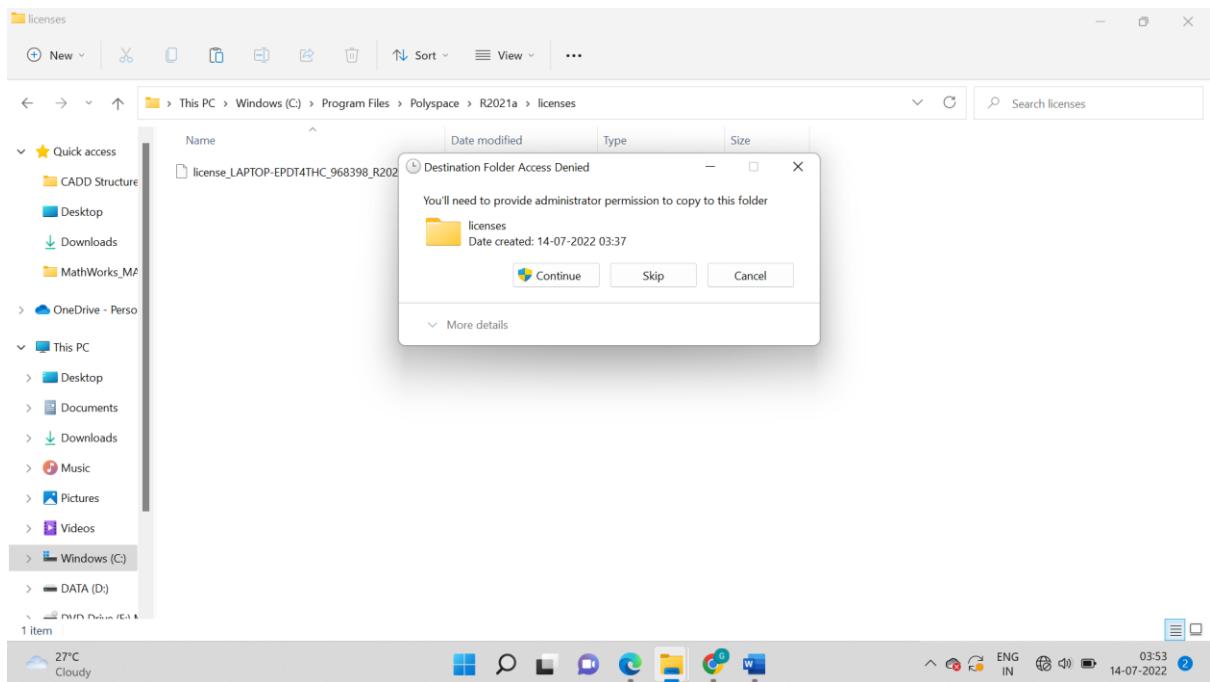




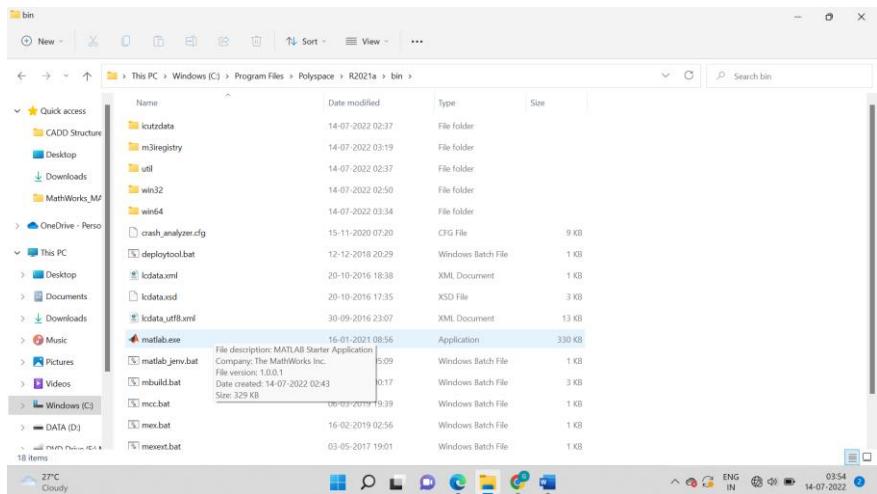
Step 14: Go to "C:\Users\Prasad\Desktop\MATLAB\MathWorks_MATLAB_R2021a_v9.10.0.1602886\libmwlmgrimpl.dll" and paste the licenses file:



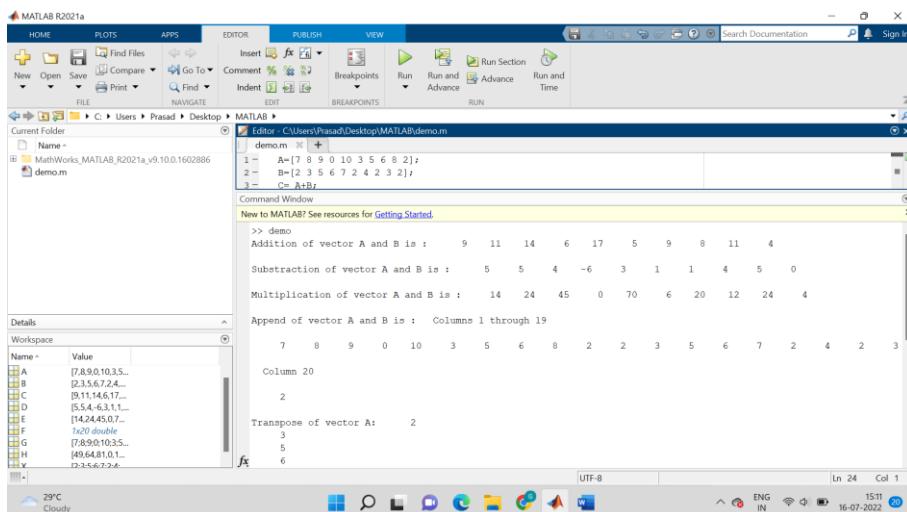
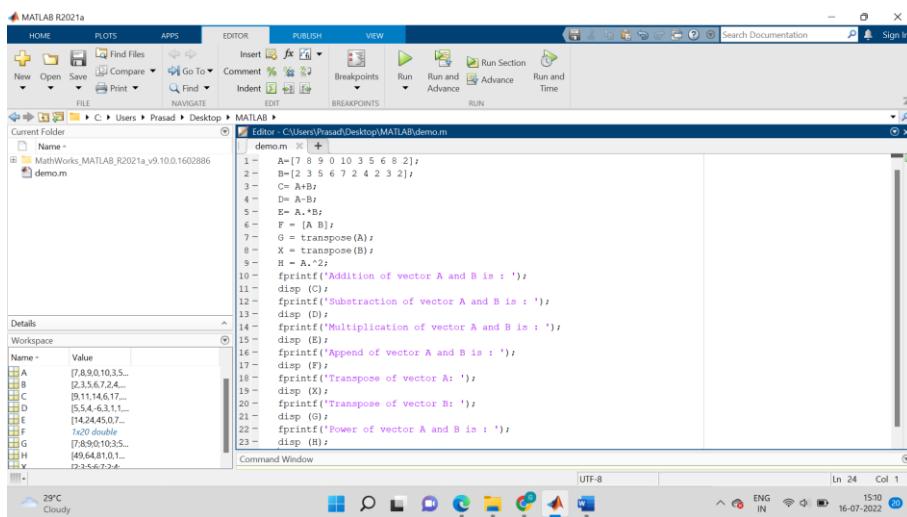




Step 15: Go to C:\Program Files\Polyspace\R2021a\bin and open the matlab.exe application:



Step 16: Try out some problems and save the file in .m format and run it:



1-D and 2-D Arrays

Vectors of MATLAB:

Vectors of MATLAB:

1. $A = [2 \ 5 \ 1 \ 6 \ 7 \ 3 \ 9 \ 10]$, add 3 to the odd indexed numbers.

Explanation:

So, we have to create a row vector now, using parenthesis and colon operator access the index required that and add 3.

Code:

```
MATLAB ▶
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m x demoo.m x +
1 - A = [1 2 3 4 5 6 7 8 9]
2 - A(1:2:end) = A(1:2:end) + 3
```

Output:

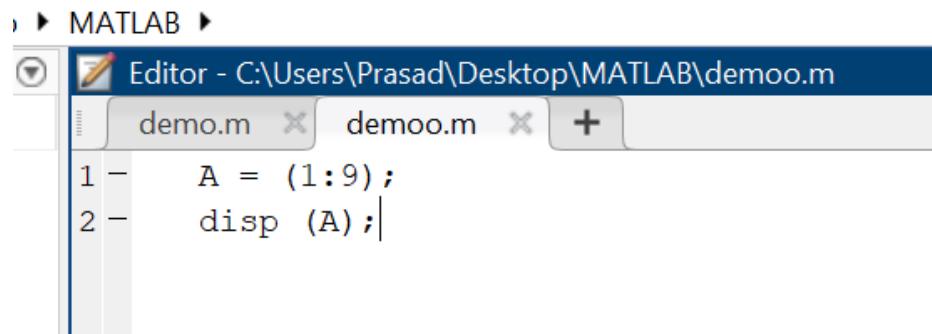
```
Command Window
New to MATLAB? See resources for Getting Started.
>> demoo
A =
    1     2     3     4     5     6     7     8     9
A =
    4     2     6     4     8     6    10     8    12
fx >>
UTF-8
```

2. Create a vector of row vector of size 9.

Explanation:

So, I have used colon notation and a row vector of range 1 to 9 is created.

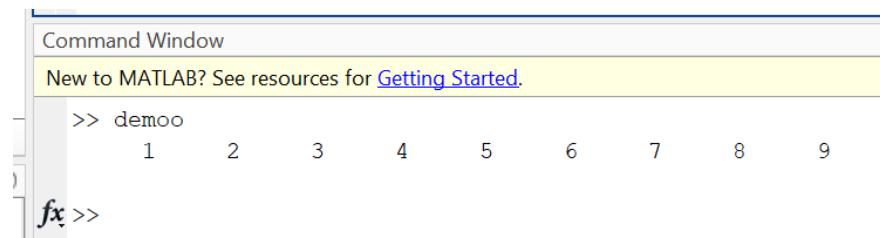
Code:



The screenshot shows the MATLAB Editor window. The title bar says "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". There are two tabs: "demo.m" and "demoo.m", with "demoo.m" being the active tab. The code in the editor is:

```
1 A = (1:9);
2 disp (A);
```

Output:



The screenshot shows the MATLAB Command Window. The title bar says "Command Window". The window displays the following text:

```
>> demoo
      1   2   3   4   5   6   7   8   9
fx >>
```

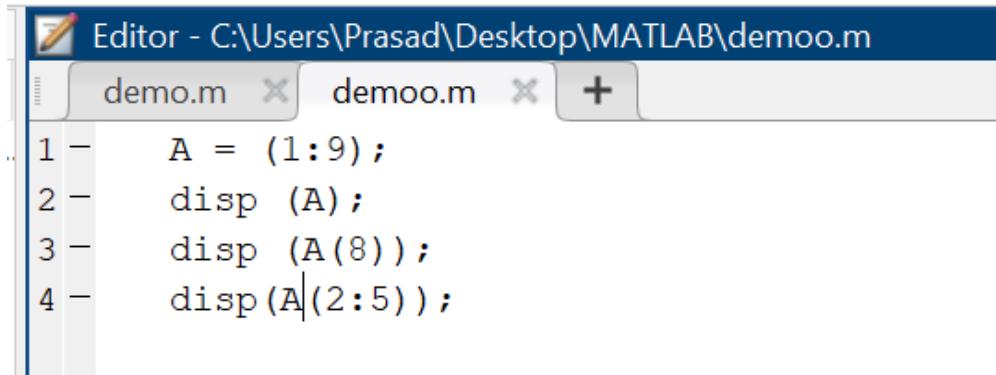
3. From the above array access the following elements:

- i. Any one element of a specific position.

Explanation:

So by using colon notation I created a row vector of range 1 to 9 is created and now by using parenthesis just access the required elements.

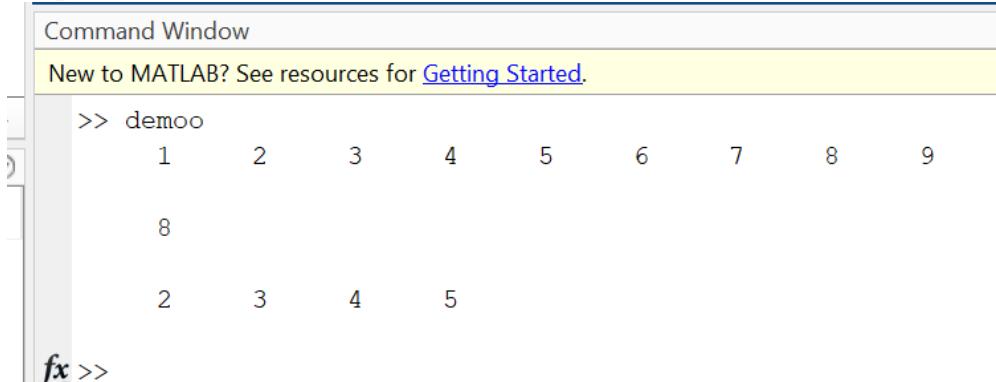
Code:



The screenshot shows the MATLAB Editor window. The title bar says "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". There are two tabs: "demo.m" and "demoo.m". The "demoo.m" tab is active, showing the following MATLAB code:

```
1 A = (1:9);
2 disp (A);
3 disp (A(8));
4 disp (A(2:5));
```

Output:



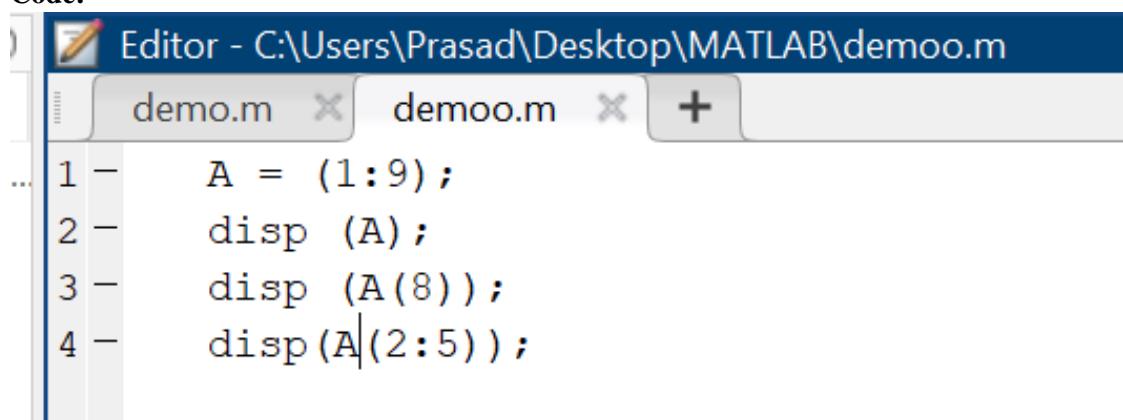
The screenshot shows the MATLAB Command Window. The title bar says "Command Window". It displays the following text:

New to MATLAB? See resources for [Getting Started](#).

```
>> demoo
1     2     3     4     5     6     7     8     9
8
2     3     4     5
```

ii. Range of any four elements.

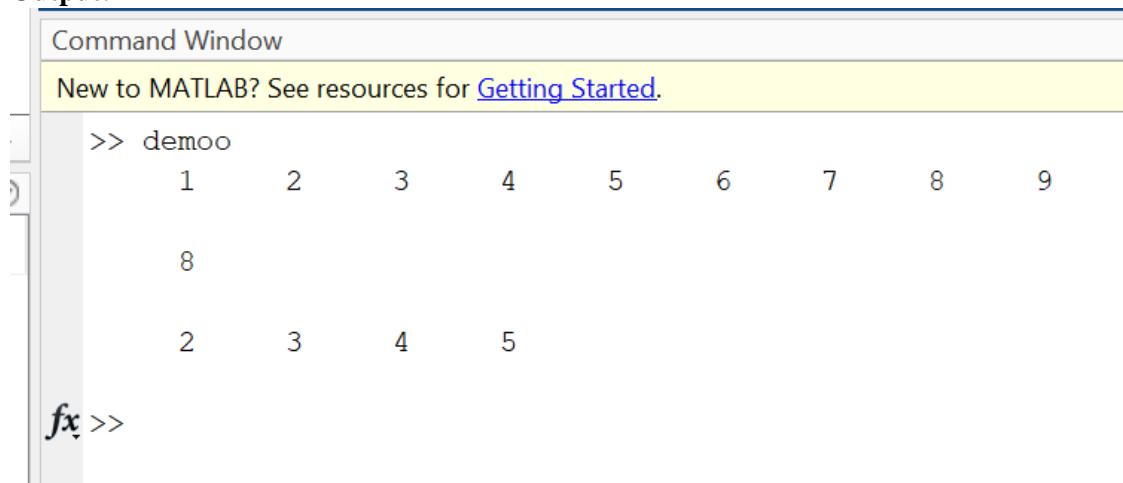
Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m  demoo.m  +
```

```
1 - A = (1:9);
2 - disp (A);
3 - disp (A(8));
4 - disp (A(2:5));
```

Output:



```
Command Window
New to MATLAB? See resources for Getting Started.
```

```
>> demoo
1     2     3     4     5     6     7     8     9
8
2     3     4     5
fx >>
```

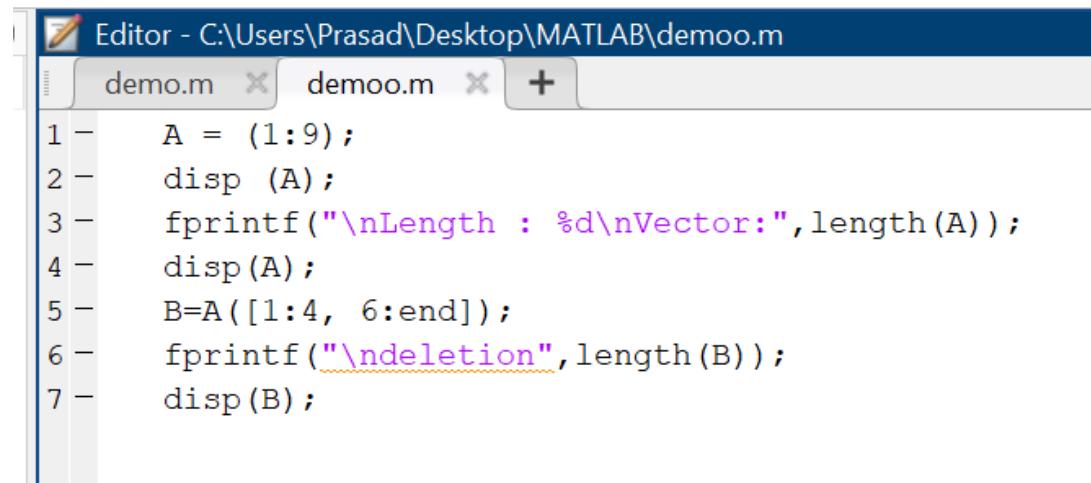
4. From the array from question 1. delete the any one element and determine the length and size of the array after deletion.

Explanation:

So by using colon notation I have created a row vector of range 1 to 9 is created now just one another vector is created excluding the element that is deleted now the function length() is used to just calculate the number of elements present in the vector.

Code:

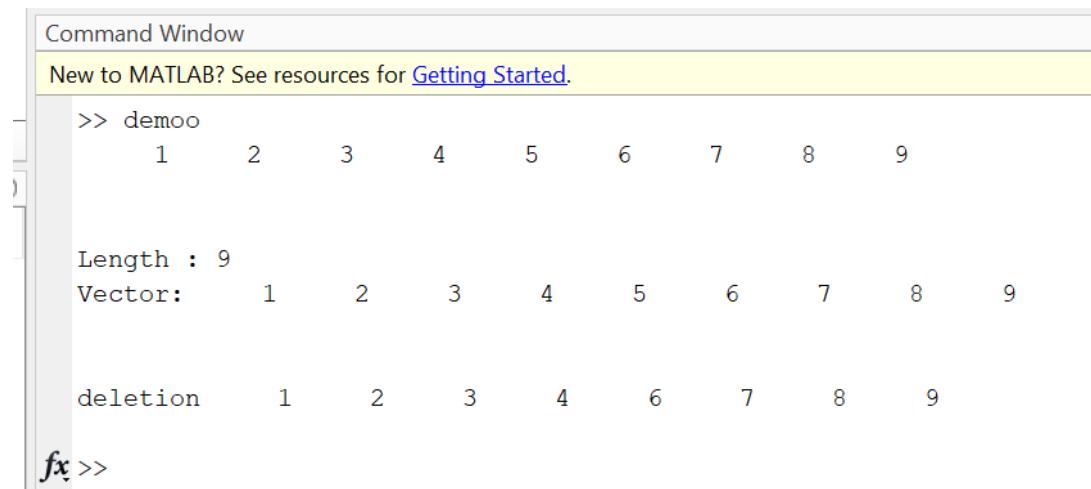
► MATLAB ►



The screenshot shows the MATLAB Editor window. The current file is 'demoo.m'. The code inside the file is as follows:

```
1 A = (1:9);
2 disp (A);
3 fprintf("\nLength : %d\nVector:",length(A));
4 disp(A);
5 B=A([1:4, 6:end]);
6 fprintf("\ndeletion",length(B));
7 disp(B);
```

Output:



The screenshot shows the MATLAB Command Window. The user has run the script 'demoo.m'. The output is:

```
>> demoo
      1      2      3      4      5      6      7      8      9

Length : 9
Vector:      1      2      3      4      5      6      7      8      9

deletion      1      2      3      4      6      7      8      9

fx >>
```

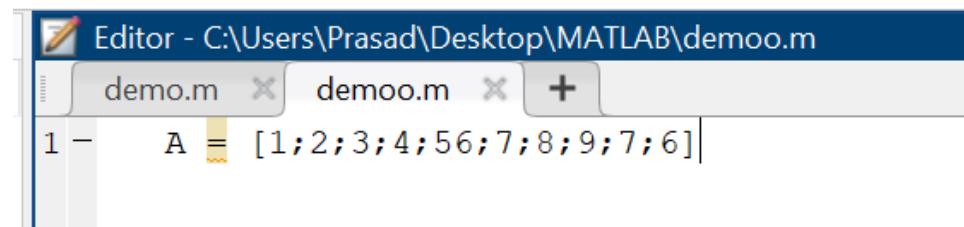
5. Create a vector of column vector of size 10.

Explanation:

So I have created a column vector by using semicolons while initialization.

Code:

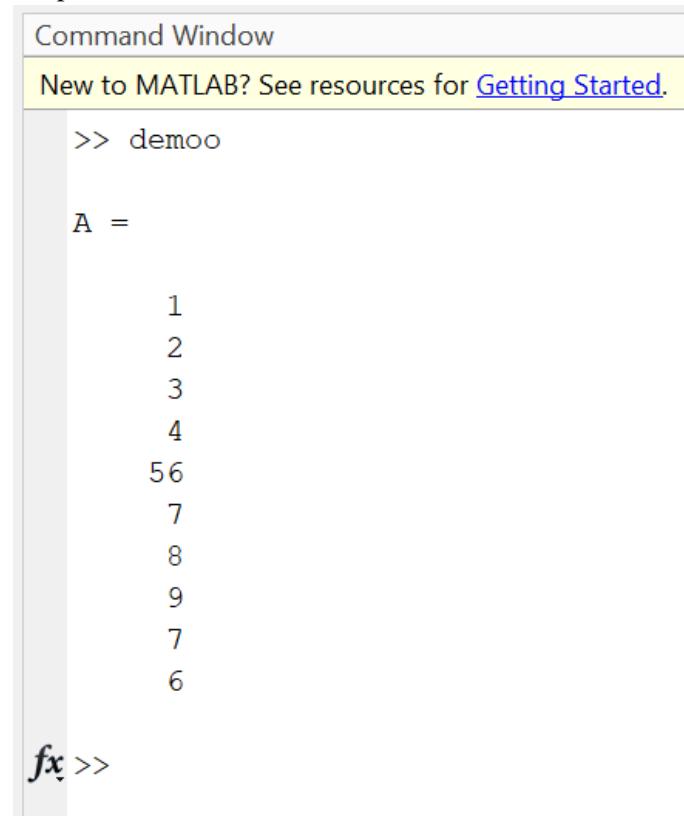
MATLAB ➔



The screenshot shows the MATLAB Editor window titled "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". It contains two tabs: "demo.m" and "demoo.m". The "demoo.m" tab is active and displays the following code:

```
1 - A = [1;2;3;4;56;7;8;9;7;6]
```

Output:



The screenshot shows the MATLAB Command Window. It starts with a welcome message: "New to MATLAB? See resources for [Getting Started](#)". Below that, the command ">> demoo" is entered, followed by the assignment "A =". The resulting output is a column vector:

```
1  
2  
3  
4  
56  
7  
8  
9  
7  
6
```

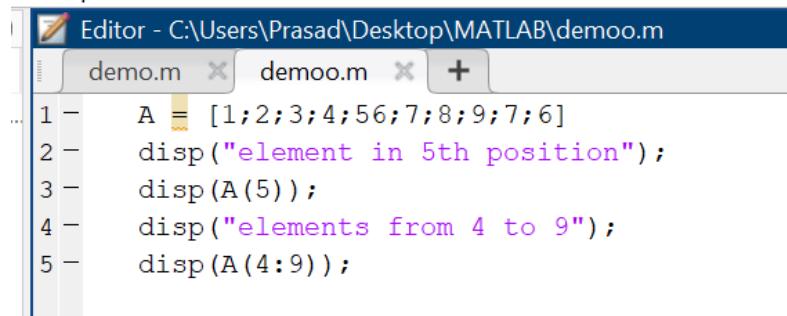
At the bottom of the window, there is a cursor icon and the text "fx >>".

6. From the above array access the following elements:
i. Any one element of a specific position.

Explanation:

So I created a column vector by using semicolons while initialization now the parenthesis and colon notations that is the range operator is used to access the needed elements.

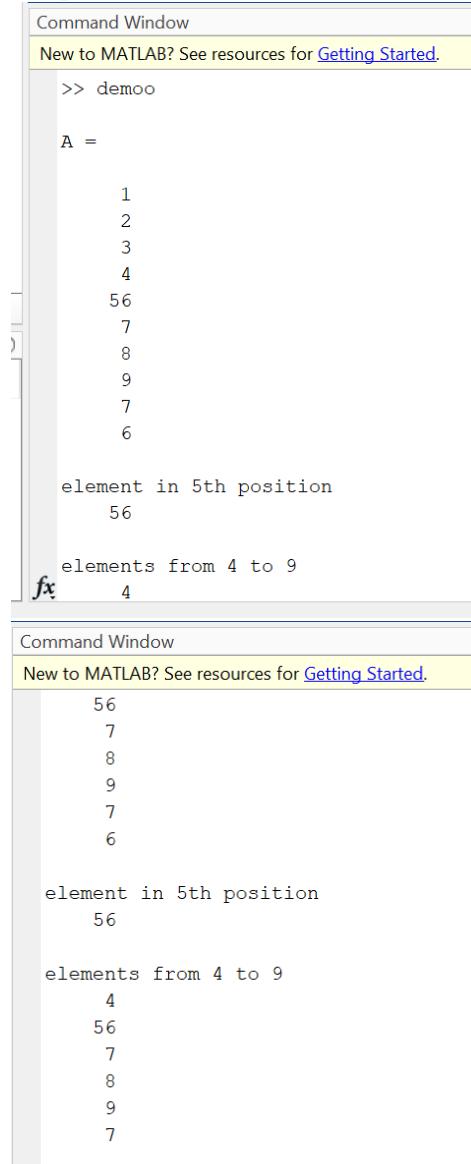
Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code defines a column vector A and uses disp() to output its elements and specific slices.

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m x demoo.m x +
1 A = [1;2;3;4;56;7;8;9;7;6]
2 disp("element in 5th position");
3 disp(A(5));
4 disp("elements from 4 to 9");
5 disp(A(4:9));
```

Output:



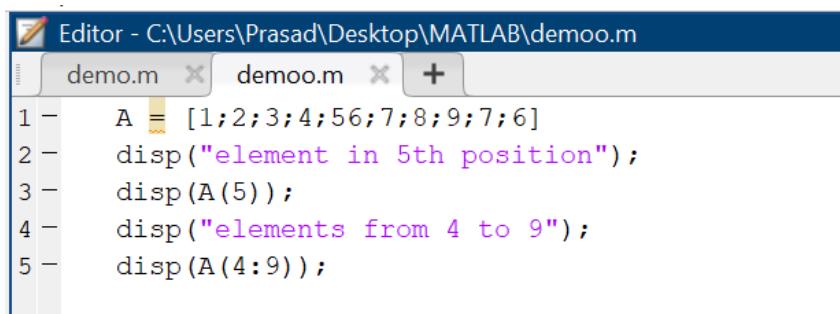
The screenshot shows the MATLAB Command Window displaying the output of running the script 'demo.m'. It shows the definition of vector A, the output of disp(A(5)), and the output of disp(A(4:9)).

```
Command Window
New to MATLAB? See resources for Getting Started.
>> demo
A =
1
2
3
4
56
7
8
9
7
6
element in 5th position
56
elements from 4 to 9
4
56
7
8
9
7
```

Command Window
New to MATLAB? See resources for [Getting Started](#).
56
7
8
9
7
6
element in 5th position
56
elements from 4 to 9
4
56
7
8
9
7

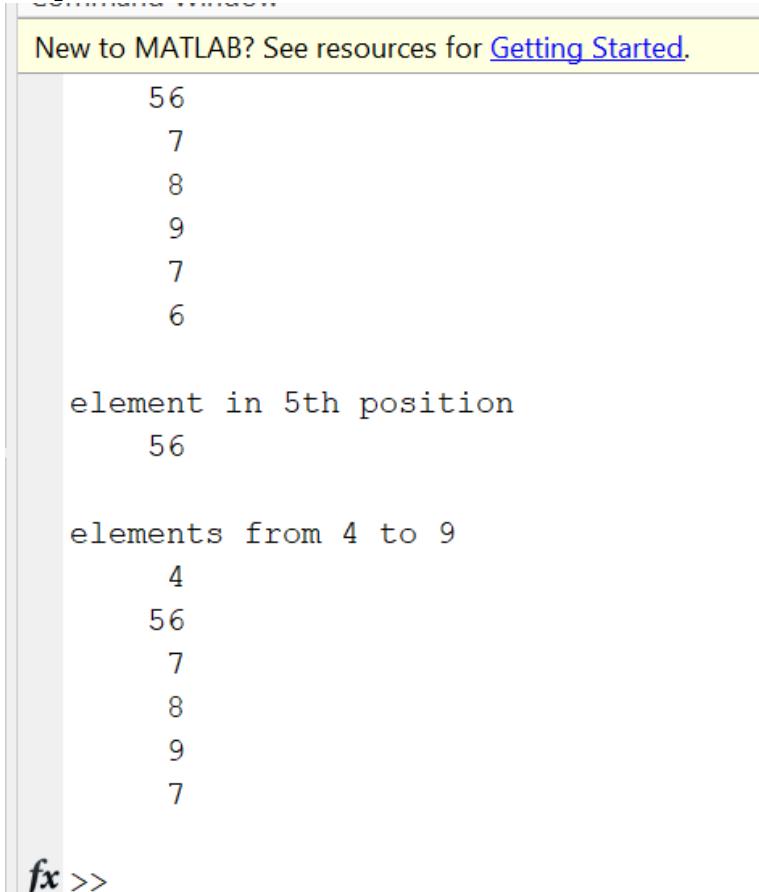
ii. Range of any four elements.

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m  demoo.m  +
1 - A = [1;2;3;4;56;7;8;9;7;6]
2 - disp("element in 5th position");
3 - disp(A(5));
4 - disp("elements from 4 to 9");
5 - disp(A(4:9));
```

Output:



```
New to MATLAB? See resources for Getting Started.
56
7
8
9
7
6

element in 5th position
56

elements from 4 to 9
4
56
7
8
9
7

fx >>
```

7. From the array from question 4. delete the any one element and determine the length and size of the array after deletion.

Explanation:

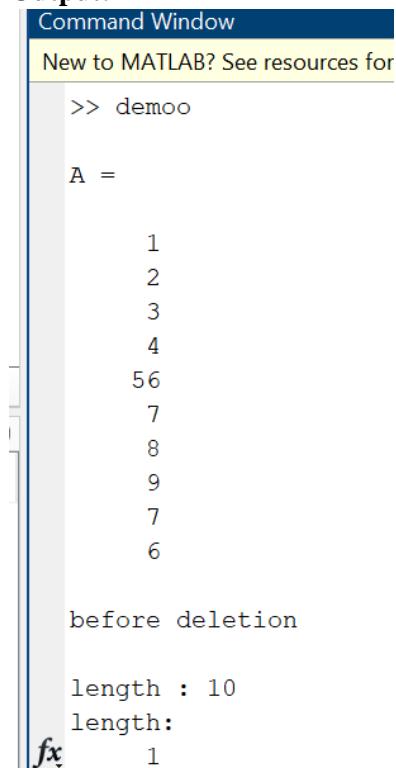
So, I have created a column vector by using semicolons and just while I was doing initialization. another vector is created that is excluding the element to be deleted now the function length () is used which calculates the number of elements present in the vector.

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m x demoo.m x +
1 - A = [1;2;3;4;56;7;8;9;7;6]
2 - disp("before deletion");
3 - fprintf("\nlength : %d\nlength:\n",length(A));
4 - disp(A);
5 - B = A([1:4, 6:end]);
6 - fprintf("after deletion",length(B));
7 - disp(B);
```

Output:



```
Command Window
New to MATLAB? See resources for
>> demoo

A =

    1
    2
    3
    4
    56
    7
    8
    9
    7
    6

before deletion

length : 10
length:
fx 1
```

Command Window

New to MATLAB? See resources for [Getting Started](#)

```
before deletion
length : 10
length:
    1
    2
    3
    4
    56
    7
    8
    9
    7
    6

after deletion      1
    2
    3
    4
    7
fx    7
```

Command Window

New to MATLAB? See resources for [Getting Started](#)

```
3
4
56
7
8
9
7
6

after deletion      1
    2
    3
    4
    7
    8|
    9
    7
    6

fx >>
```

8. $A = [2 \ 5 \ 1 \ 6 \ 7 \ 3 \ 9 \ 10 \ 11 \ 15 \ 14]$, add 3 to the odd indexed numbers.

Explanation:

So, we have to create a row vector by using parenthesis and colon operator we can access the index required and add 3 as shown:

Code:

The screenshot shows the MATLAB Editor window with the file 'demoo.m' open. The code in the editor is:

```
1 - A = [2 5 1 6 7 3 9 10 11 15 14];
2 - disp("vector before:")
3 - disp(A);
4 - A(1:2:end)=A(1:2:end)+3;
5 - disp("After addition of 3");
6 - disp(A)
```

Output:

The screenshot shows the MATLAB Command Window. The output of running the script 'demoo.m' is:

```
Command Window
New to MATLAB? See resources for Getting Started.
>> demoo

A =
1
2
3
4
56
7
8
9
7
6

before deletion

length : 10
length:
1
fx 2

Command Window
New to MATLAB? See resources for Getting Started.
before deletion

length : 10
length:
1
2
3
4
56
7
8
9
7
6

after deletion      1
2
3
fx 4
```

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
after deletion      1
2
3
4
7
8
9
7
6

>> demo0
vector before:
2      5      1      6      7      3      9      10     11     15     14

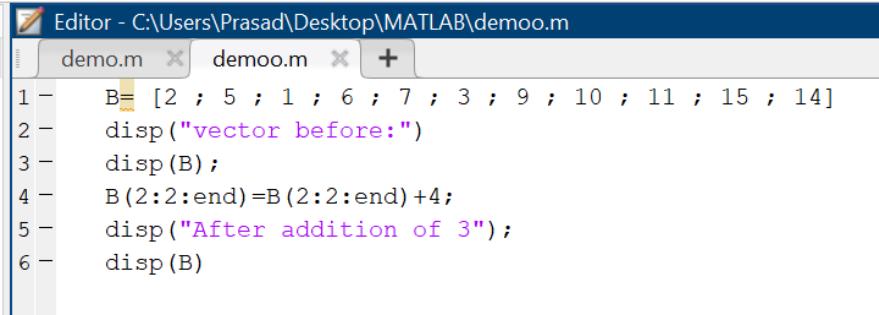
After addition of 3
5      5      4      6      10     3      12     10     14     15     17
```

9. $B = [2 ; 5 ; 1 ; 6 ; 7 ; 3 ; 9 ; 10 ; 11 ; 15 ; 14]$, add 4 to the even indexed numbers.

Explanation:

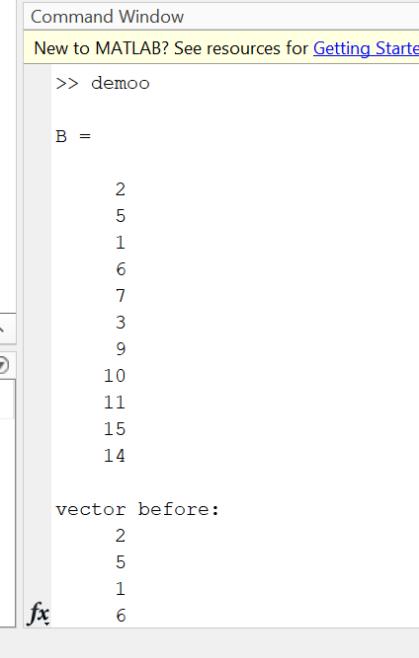
So I have created a row vector by just using parenthesis and colon operator that access the index required and add 4.

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m demoo.m + 
1 - B= [2 ; 5 ; 1 ; 6 ; 7 ; 3 ; 9 ; 10 ; 11 ; 15 ; 14]
2 - disp("vector before:")
3 - disp(B);
4 - B(2:2:end)=B(2:2:end)+4;
5 - disp("After addition of 3");
6 - disp(B)
```

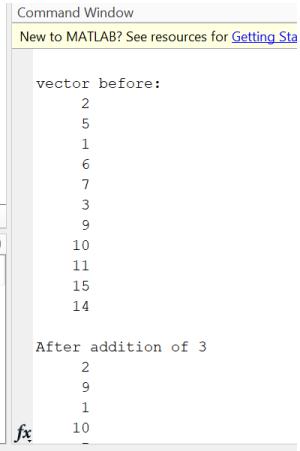
Output:



```
Command Window
New to MATLAB? See resources for Getting Started
>> demoo

B =
2
5
1
6
7
3
9
10
11
15
14

vector before:
2
5
1
6
```



```
Command Window
New to MATLAB? See resources for Getting Started

vector before:
2
5
1
6
7
3
9
10
11
15
14

After addition of 3
2
9
1
10
```

Command Window

New to MATLAB? See resources for [Getting Started](#)

```
    9
    10
    11
    15
    14

After addition of 3
    2
    9
    1
    10
    7
    7
    9
    14
    11
    19
    14

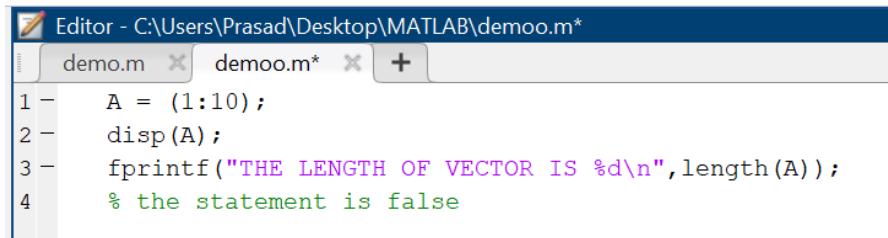
f~.~.
```

10. If the vector is of size 1 X 10, then the length of the vector will be more than 10. Justify whether the statement is true or false with the help of an example.

Explanation:

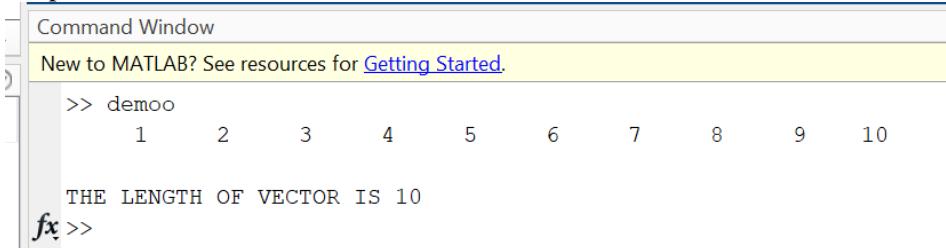
So I have created a row vector from range 1 to 10 As we know a range function take both the start and end point into consideration, so a vector of length no less than 10 is created.

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m*
demo.m  demo.m*  +
1 - A = (1:10);
2 - disp(A);
3 - fprintf("THE LENGTH OF VECTOR IS %d\n",length(A));
4 - % the statement is false
```

Output:



```
Command Window
New to MATLAB? See resources for Getting Started.
>> demo
    1     2     3     4     5     6     7     8     9     10
THE LENGTH OF VECTOR IS 10
fx >>
```

Matrices of MATLAB:

1. Create a matrix of size 5.

Explanation:

A magic matrix is created that is of dimension 5x5.

Code:

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m x demoo.m x + 
1 - A = magic(5);
2 - disp(A)
3 - disp("ELEMENTS IN FIRST ROW AND SECOND COLUMN IS:")
4 - disp(A(1,4));
```

Output:

```
Command Window
New to MATLAB? See resources for Getting Started.
>> demoo
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

ELEMENTS IN FIRST ROW AND SECOND COLUMN IS:
    8

Range of any 3 elements in 3rd column:
    1
    7
   13

All elements in row 2:
    5     7    14    16
fx
```

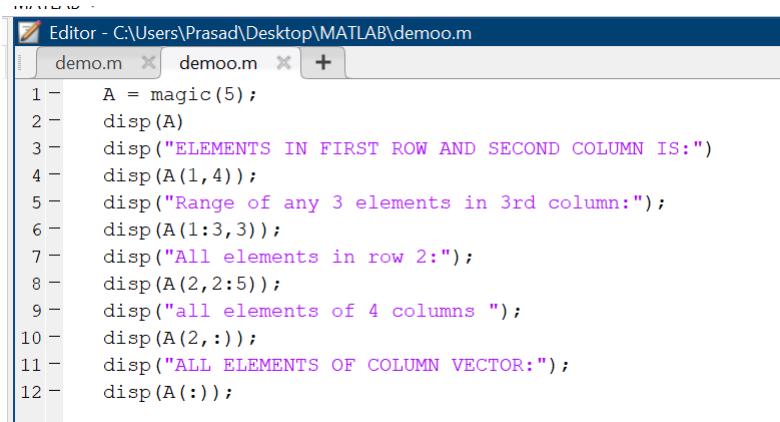
2. From the above array access the following elements:

- i. Any one element of a specific position.
- ii. Range of any four elements in a row.
- iii. Range of any three elements in a column.
- iv. All the elements of two rows.
- v. All the elements of any 4 columns.
- vi. All the elements in a column vector.

Explanation:

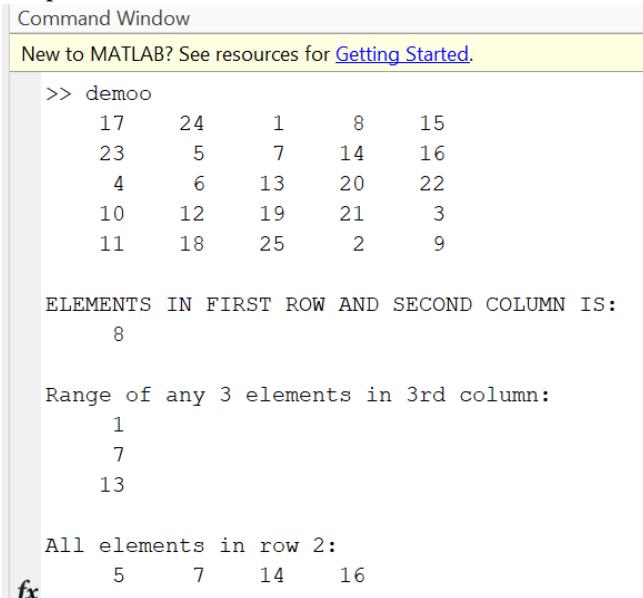
So, I have created a magic matrix that is of dimension 5x5 also , we have parenthesis and range operator (:) that we can use to access the required elements.

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m x demoo.m x +
1 - A = magic(5);
2 - disp(A)
3 - disp("ELEMENTS IN FIRST ROW AND SECOND COLUMN IS:")
4 - disp(A(1,4));
5 - disp("Range of any 3 elements in 3rd column:");
6 - disp(A(1:3,3));
7 - disp("All elements in row 2:");
8 - disp(A(2,2:5));
9 - disp("all elements of 4 columns ");
10 - disp(A(2,:));
11 - disp("ALL ELEMENTS OF COLUMN VECTOR:");
12 - disp(A(:));
```

Output:



```
Command Window
New to MATLAB? See resources for Getting Started.
>> demoo
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

ELEMENTS IN FIRST ROW AND SECOND COLUMN IS:
    8

Range of any 3 elements in 3rd column:
    1
    7
   13

All elements in row 2:
    5     7    14    16
fx
```

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
ELEMENTS IN FIRST ROW AND SECOND COLUMN IS:  
8  
  
Range of any 3 elements in 3rd column:  
1  
7  
13  
  
All elements in row 2:  
5 7 14 16  
  
all elements of 4 columns  
23 5 7 14 16  
  
ALL ELEMENTS OF COLUMN VECTOR:  
17  
23
```

fx

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
ALL ELEMENTS OF COLUMN VECTOR:  
17  
23  
4  
10  
11  
24  
5  
6  
12  
18  
1  
7  
13  
19  
25  
8  
14
```

fx

Command Window

New to MATLAB? See resources for [Getting](#).

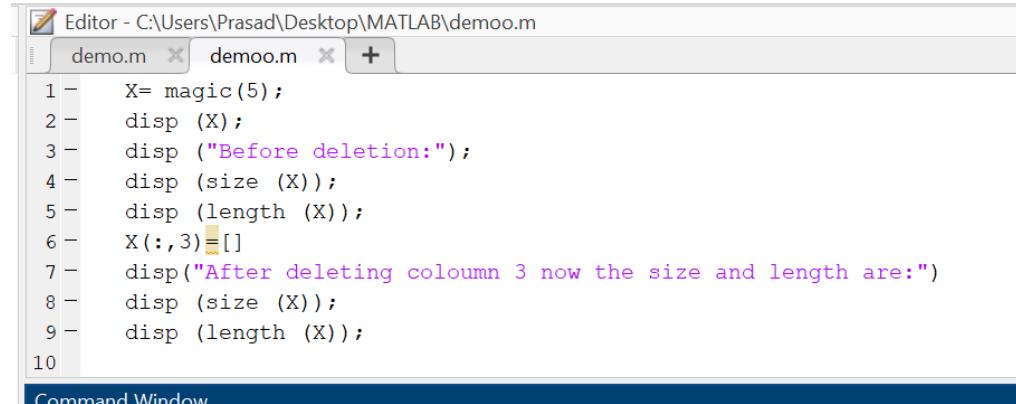
```
12  
18  
1  
7  
13  
19  
25  
8  
14  
20  
21  
2  
15  
16  
22  
3  
9
```

3. From the array from question 1. delete the any one column and determine the length and size of the array after deletion.

Explanation:

So, I have created a matrix of dimension 5x5 now, parenthesis and range operator (:) is used that will access the required element and declared it as empty.

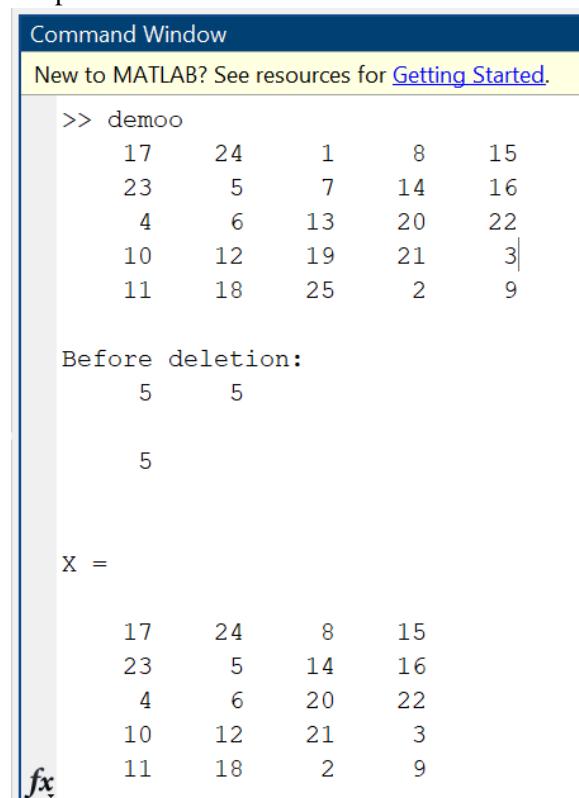
Code:



The screenshot shows the MATLAB Editor window with two tabs: 'demo.m' and 'demoo.m'. The 'demo.m' tab is active, displaying the following code:

```
1 - X= magic(5);
2 - disp (X);
3 - disp ("Before deletion:");
4 - disp (size (X));
5 - disp (length (X));
6 - X(:,3)=[];
7 - disp("After deleting coloumn 3 now the size and length are:")
8 - disp (size (X));
9 - disp (length (X));
10
```

Output:



The screenshot shows the MATLAB Command Window with the title 'Command Window' and the message 'New to MATLAB? See resources for [Getting Started](#)'. The window displays the output of running the 'demo.m' script:

```
>> demo
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

Before deletion:
    5     5
    5

X =
    17    24     8    15
    23     5    14    16
     4     6    20    22
    10    12    21     3
    11    18     2     9
```

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
Before deletion:
 5      5

 5

X =
 17    24     8    15
 23     5    14    16
  4     6    20    22
 10    12    21     3
 11    18     2     9

After deleting column 3 now the size and length are:
 5      4

 5|
```

f1x >>

4. Create a 2-D array of size of 4 X 5 to perform the following operations:

Explanation:

So I have created two matrices and concatenated accordingly by using [].

- i. Concatenate Horizontally
- ii. Concatenate Vertically

Code:

The screenshot shows the MATLAB Editor window with the file 'demoo.m' open. The code in the editor is:

```
1 - A = ones(4,5);
2 - B = zeros(4,5);
3 - fprintf("Concatenate Horizontally \n");
4 - disp([A B]);
5 - fprintf("Concatenate Vertically\n");
6 - disp([A;B]);
```

Output:

The screenshot shows the MATLAB Command Window. The command 'demoo' was run, and the output is:

```
>> demoo
Concatenate Horizontally
1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0

Concatenate Vertically
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

The screenshot shows the MATLAB Command Window again. The command 'demoo' was run, and the output is identical to the previous one:

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

Concatenate Vertically
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

Multi-Dimensional Arrays and Colon Notation

Arrays of MATLAB:

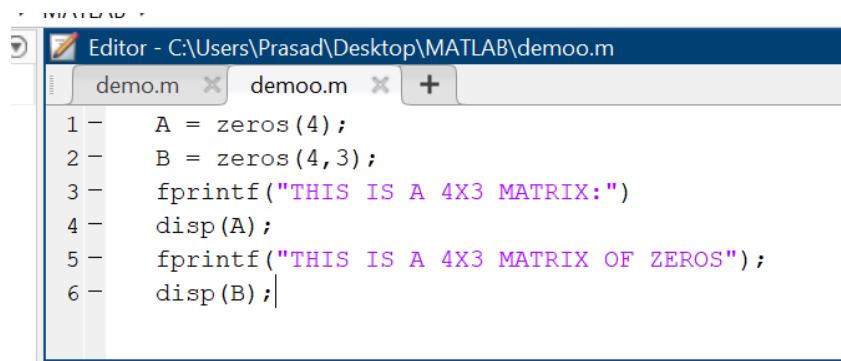
1. Create the following 2-D arrays.

Explanation:

All these functions are used to create the 2D arrays.

- a. Arrays containing 0s as elements of size 4 and 4X3.

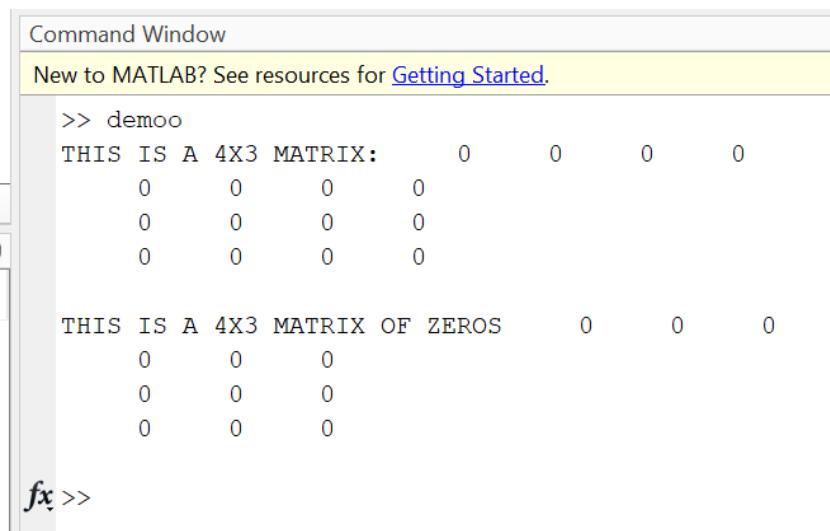
Code:



The screenshot shows the MATLAB Editor window with two files open: 'demo.m' and 'demoo.m'. The 'demo.m' file contains the following code:

```
1 - A = zeros(4);
2 - B = zeros(4, 3);
3 - fprintf("THIS IS A 4X3 MATRIX:")
4 - disp(A);
5 - fprintf("THIS IS A 4X3 MATRIX OF ZEROS");
6 - disp(B);
```

Output:



The screenshot shows the MATLAB Command Window. The user runs the command `>> demo`. The output displays two matrices. The first matrix, labeled 'THIS IS A 4X3 MATRIX:', is a 4x3 matrix of zeros:

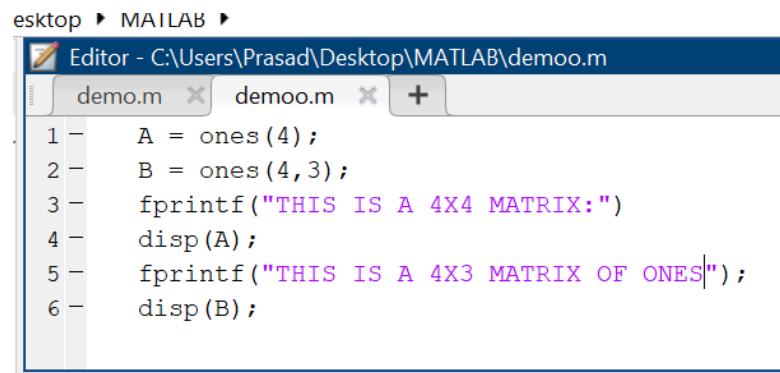
0	0	0
0	0	0
0	0	0
0	0	0

The second matrix, labeled 'THIS IS A 4X3 MATRIX OF ZEROS', is also a 4x3 matrix of zeros:

0	0	0
0	0	0
0	0	0
0	0	0

b. Arrays containing 1s as elements of size 4 and 4X3.

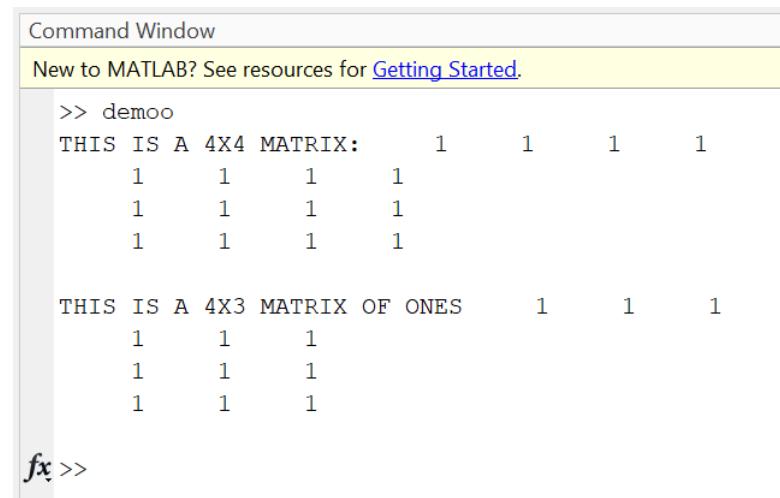
Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code in the editor is as follows:

```
1 - A = ones(4);
2 - B = ones(4, 3);
3 - fprintf("THIS IS A 4X4 MATRIX:")
4 - disp(A);
5 - fprintf("THIS IS A 4X3 MATRIX OF ONES");
6 - disp(B);
```

Output:



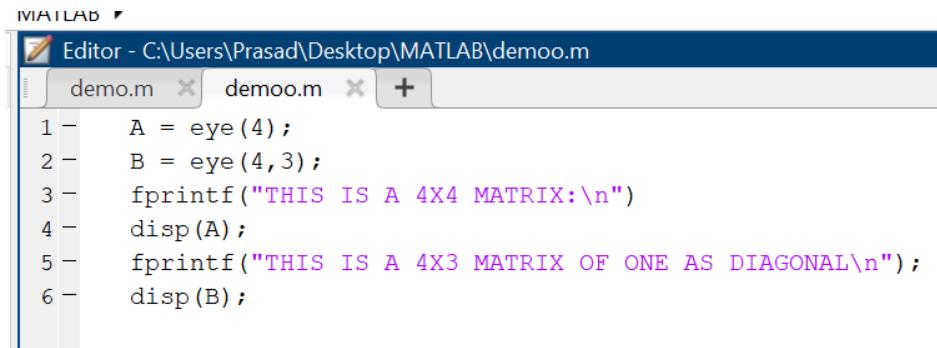
The screenshot shows the MATLAB Command Window. The user has run the script 'demo.m'. The output is:

```
>> demo
THIS IS A 4X4 MATRIX:
1 1 1 1
1 1 1 1
1 1 1 1

THIS IS A 4X3 MATRIX OF ONES
1 1 1
1 1 1
1 1 1
```

c. Arrays containing 1s (as diagonal elements) and 0s as elements of size 4 and 4X3.

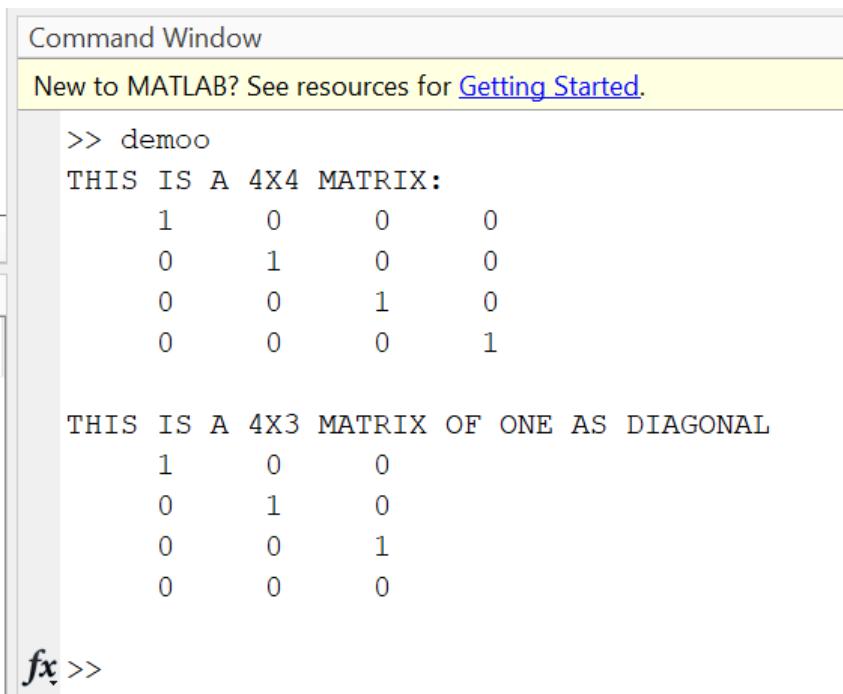
Code:



The screenshot shows the MATLAB Editor interface. The title bar says "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". There are two tabs open: "demo.m" and "demoo.m". The "demo.m" tab contains the following code:

```
1 - A = eye(4);
2 - B = eye(4,3);
3 - fprintf("THIS IS A 4X4 MATRIX:\n")
4 - disp(A);
5 - fprintf("THIS IS A 4X3 MATRIX OF ONE AS DIAGONAL\n");
6 - disp(B);
```

Output:



The screenshot shows the MATLAB Command Window. It starts with a welcome message: "New to MATLAB? See resources for [Getting Started](#)". Then it executes the command ">> demoo". The output consists of two parts. The first part is the 4x4 matrix A:

```
THIS IS A 4X4 MATRIX:
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

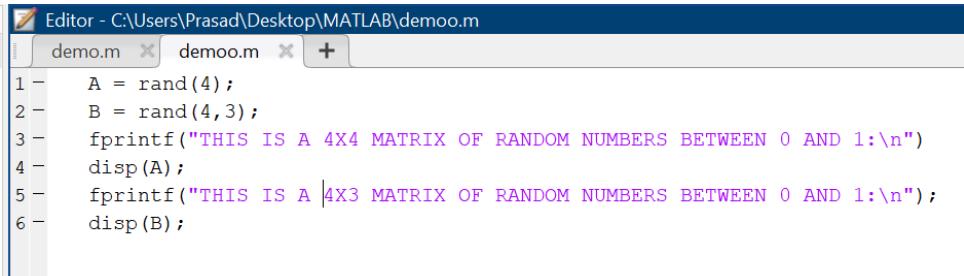
The second part is the 4x3 matrix B:

```
THIS IS A 4X3 MATRIX OF ONE AS DIAGONAL
1 0 0
0 1 0
0 0 1
0 0 0
```

At the bottom left, there is a small icon labeled "fx" followed by ">>".

d. Array containing random elements from 0-1 of size 4 and 4X3.

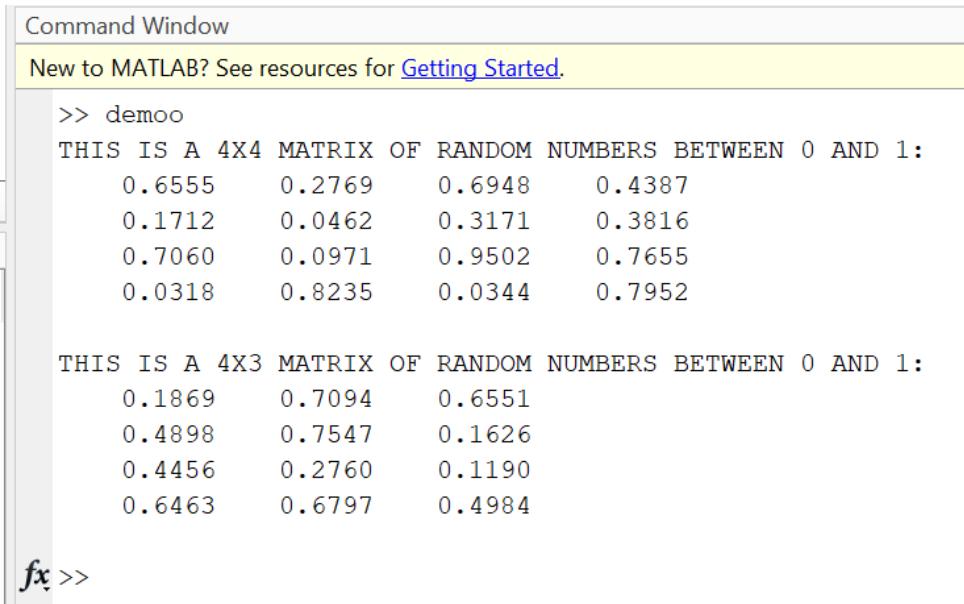
Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code in the editor is as follows:

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m demoo.m +
1- A = rand(4);
2- B = rand(4, 3);
3- fprintf("THIS IS A 4X4 MATRIX OF RANDOM NUMBERS BETWEEN 0 AND 1:\n")
4- disp(A);
5- fprintf("THIS IS A |4X3 MATRIX OF RANDOM NUMBERS BETWEEN 0 AND 1:\n");
6- disp(B);
```

Output:



The screenshot shows the MATLAB Command Window. The user has run the script 'demo.m'. The output displays two matrices of random numbers between 0 and 1.

```
Command Window
New to MATLAB? See resources for Getting Started.
>> demo
THIS IS A 4X4 MATRIX OF RANDOM NUMBERS BETWEEN 0 AND 1:
0.6555    0.2769    0.6948    0.4387
0.1712    0.0462    0.3171    0.3816
0.7060    0.0971    0.9502    0.7655
0.0318    0.8235    0.0344    0.7952

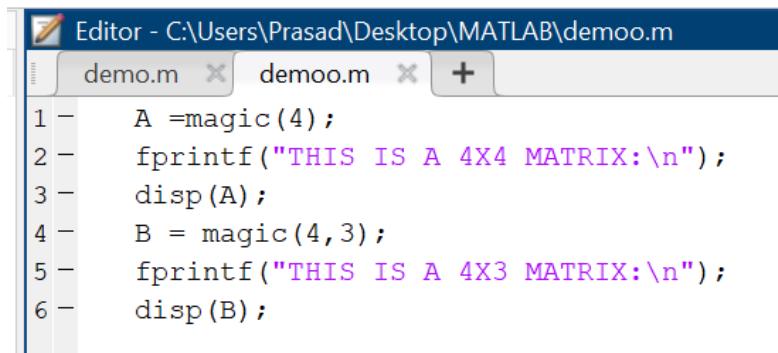
THIS IS A 4X3 MATRIX OF RANDOM NUMBERS BETWEEN 0 AND 1:
0.1869    0.7094    0.6551
0.4898    0.7547    0.1626
0.4456    0.2760    0.1190
0.6463    0.6797    0.4984
```

e. Arrays using `magic()` of size 4 and 4X3.

Magic matrix:

Magic matrix will work only when the number of rows and columns are same.

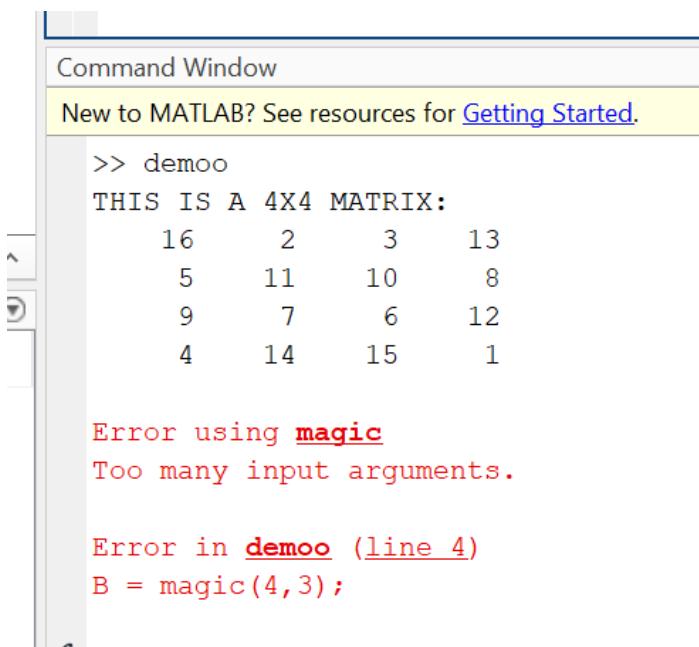
Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 A =magic(4);
2 fprintf("THIS IS A 4X4 MATRIX:\n");
3 disp(A);
4 B = magic(4,3);
5 fprintf("THIS IS A 4X3 MATRIX:\n");
6 disp(B);
```

Output:

Magic matrix will work only when the number of rows and columns are same.



Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> demoo
THIS IS A 4X4 MATRIX:
 16      2      3     13
   5     11     10      8
   9      7      6     12
   4     14     15      1

Error using magic
Too many input arguments.

Error in demoo (line 4)
B = magic(4,3);
```

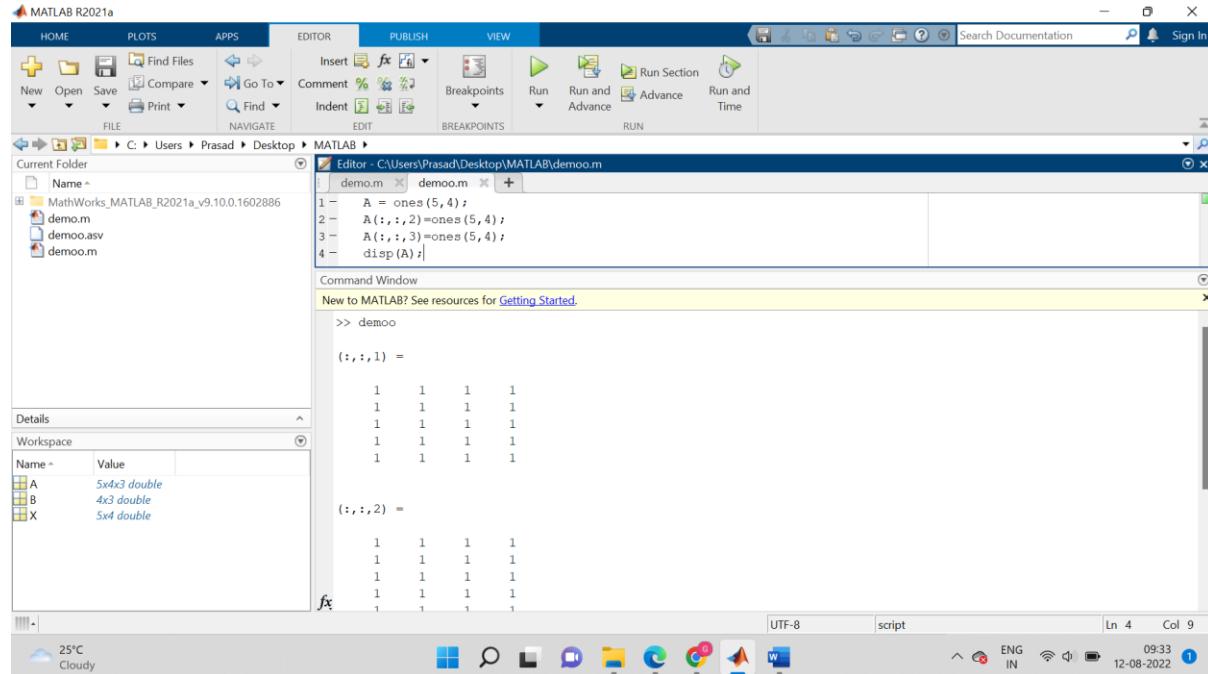
2. Create the following multidimensional arrays of size 5X4 with 3 dimensions:

Explanation:

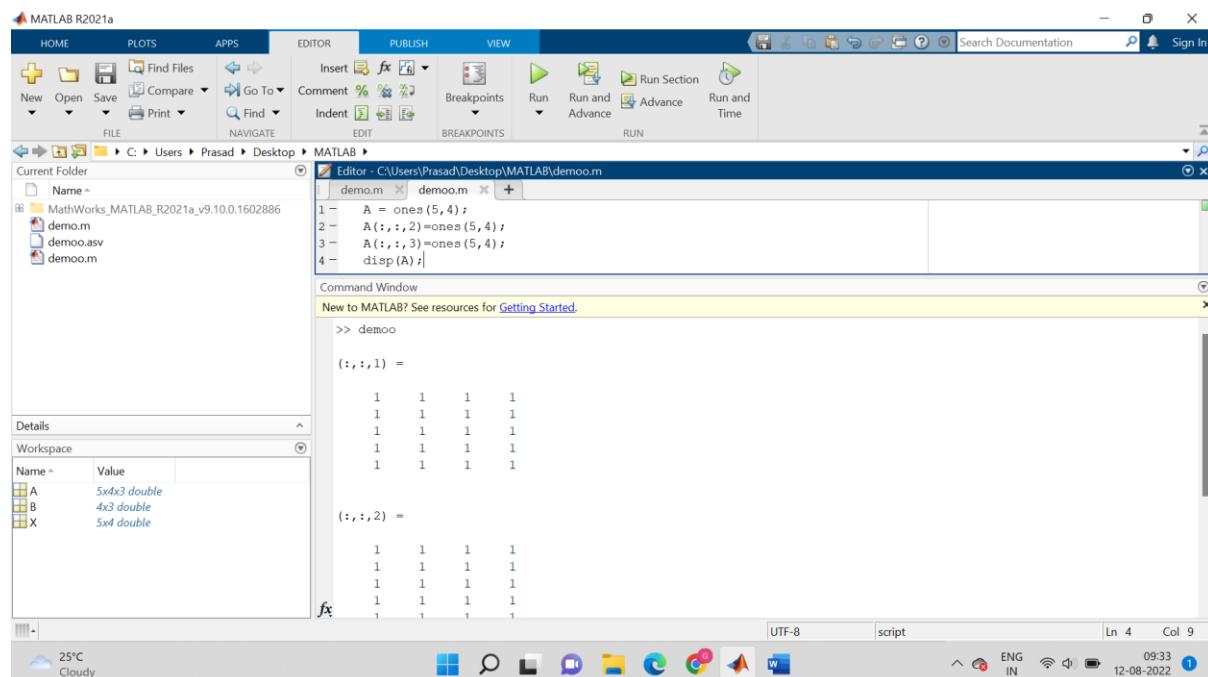
So, I have created an array of required dimension. Now the next dimensions is added, one by one to make it a multidimensional array.

i) `ones()`

Code:



Output:



The screenshot shows the MATLAB R2021a interface. The Editor tab is selected, displaying the code for demo.m:

```

1 - A = ones(5,4);
2 - A(:,:,2)=ones(5,4);
3 - A(:,:,3)=ones(5,4);
4 - disp(A);

```

The Command Window shows the output of the disp(A) command:

```

(:,:,1) =
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

(:,:,2) =
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

```

The Workspace browser shows variables A, B, and X defined as double arrays.

ii) zeros()

Code:

The screenshot shows the MATLAB R2021a interface. The Editor tab is selected, displaying the code for demo.m:

```

1 - A = zeros(5,4);
2 - A(:,:,2)=zeros(5,4);
3 - A(:,:,3)=zeros(5,4);
4 - disp(A);

```

The Command Window shows the output of the disp(A) command:

```

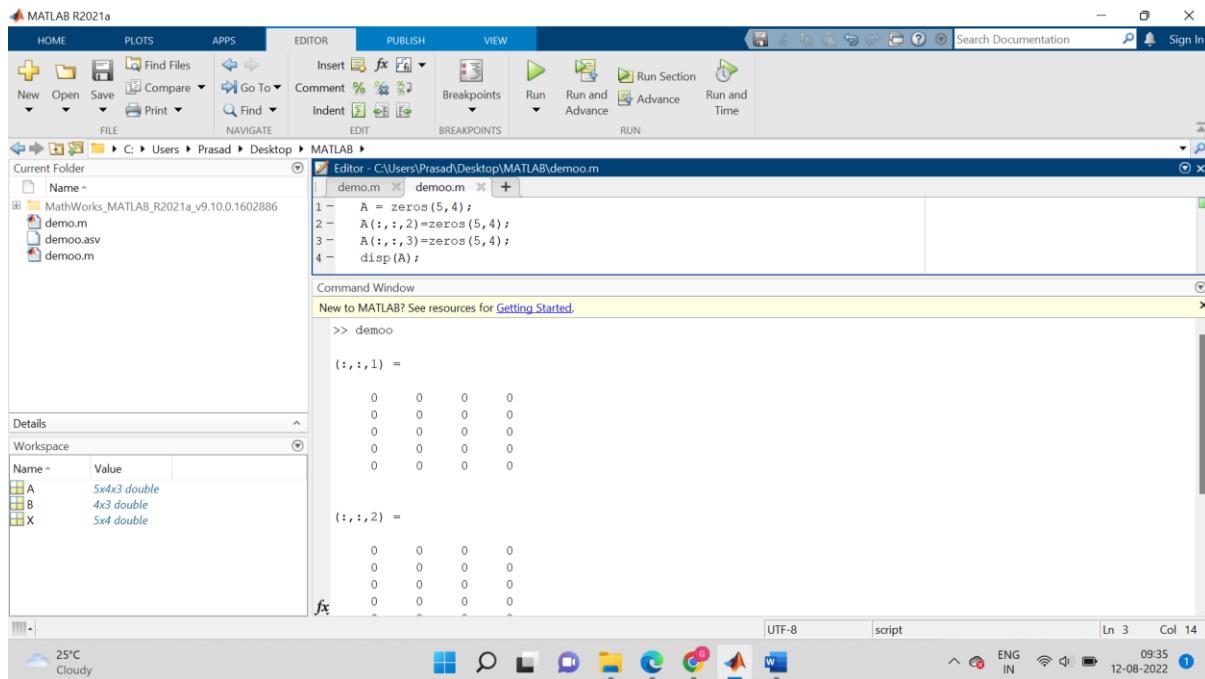
(:,:,1) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

(:,:,2) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

```

The Workspace browser shows variables A, B, and X defined as double arrays.

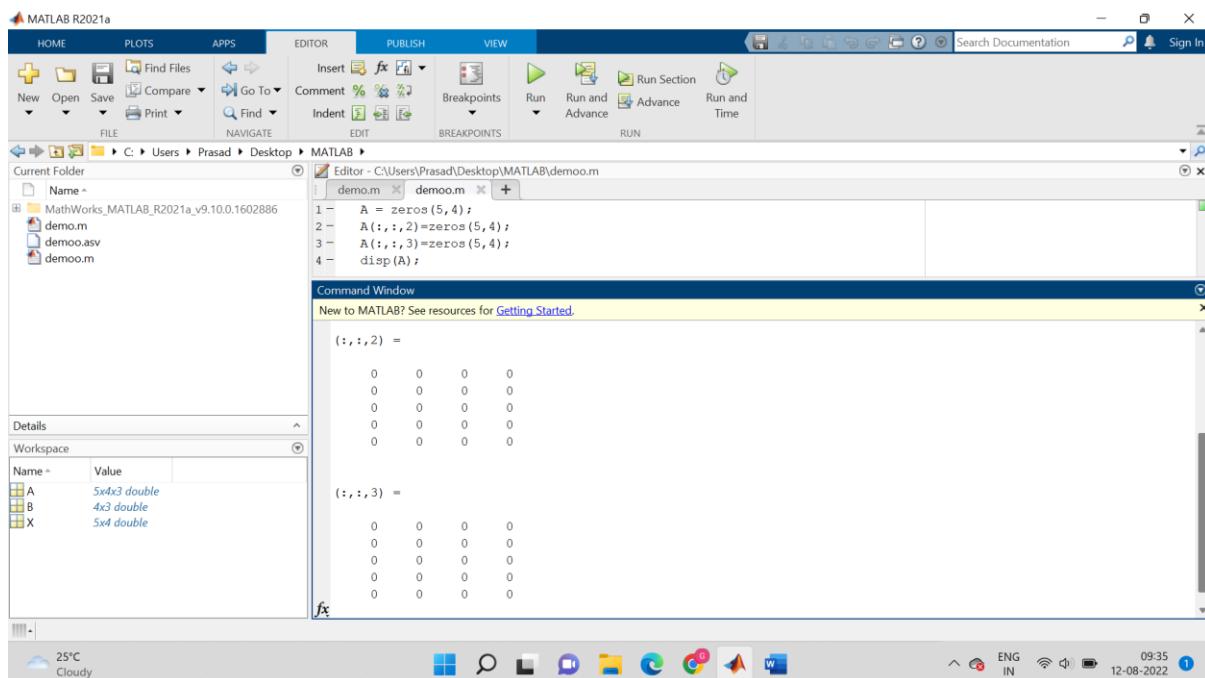
Output:



The screenshot shows the MATLAB R2021a interface. The command window displays the execution of the script `demo.m`. The output shows two 5x4 matrices, `(:,:,1)` and `(:,:,2)`, both of which are zero matrices.

```
>> demo
(:,:,1) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

(:,:,2) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```



The screenshot shows the MATLAB R2021a interface. The command window displays the execution of the script `demo.m`. The output shows three 5x4 matrices, `(:,:,1)`, `(:,:,2)`, and `(:,:,3)`, all of which are zero matrices.

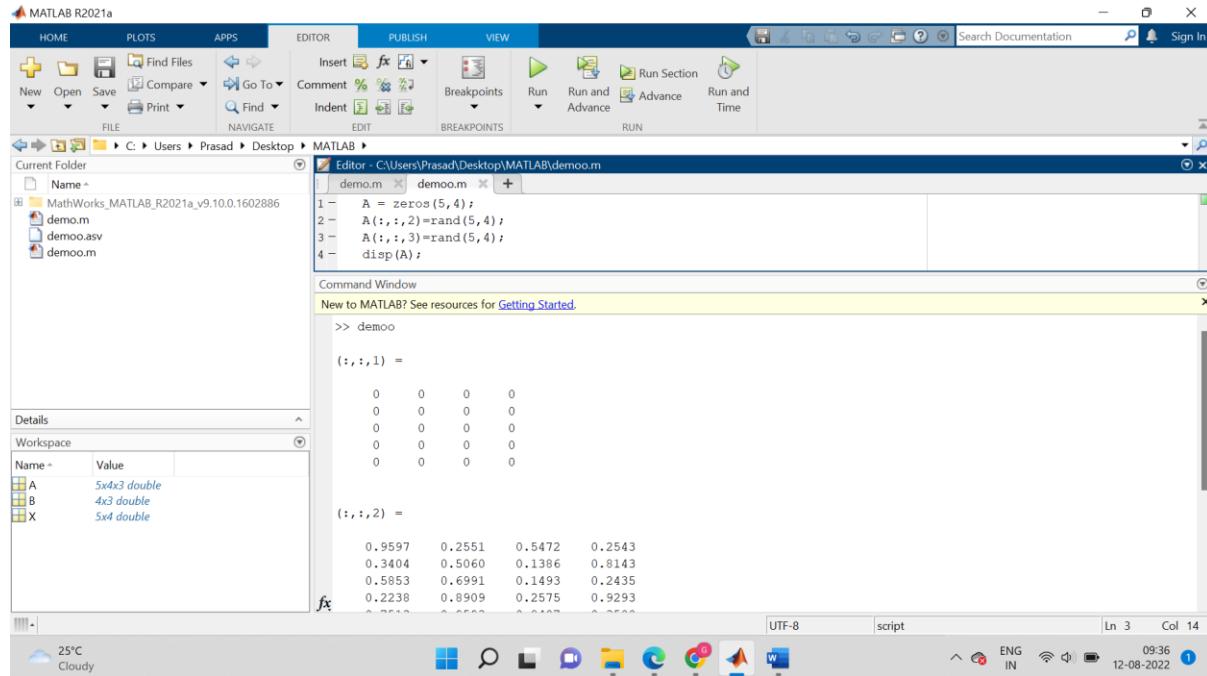
```
>> demo
(:,:,1) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

(:,:,2) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

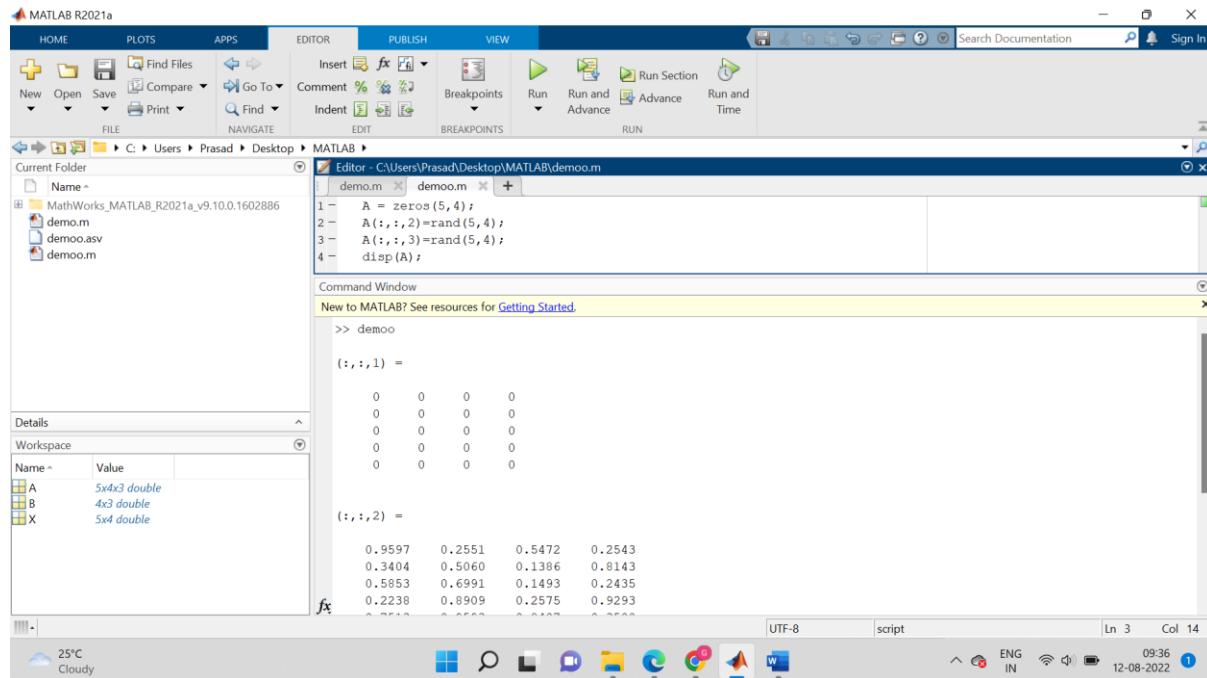
(:,:,3) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

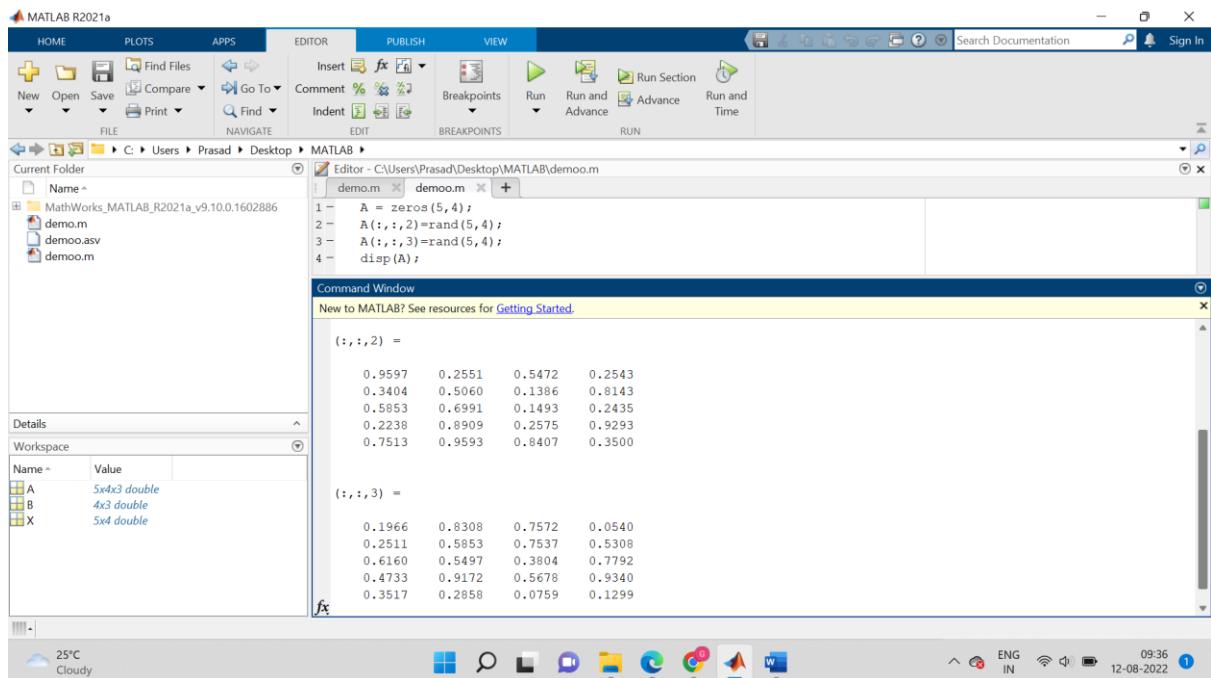
iii) rand()

Code:



Output:



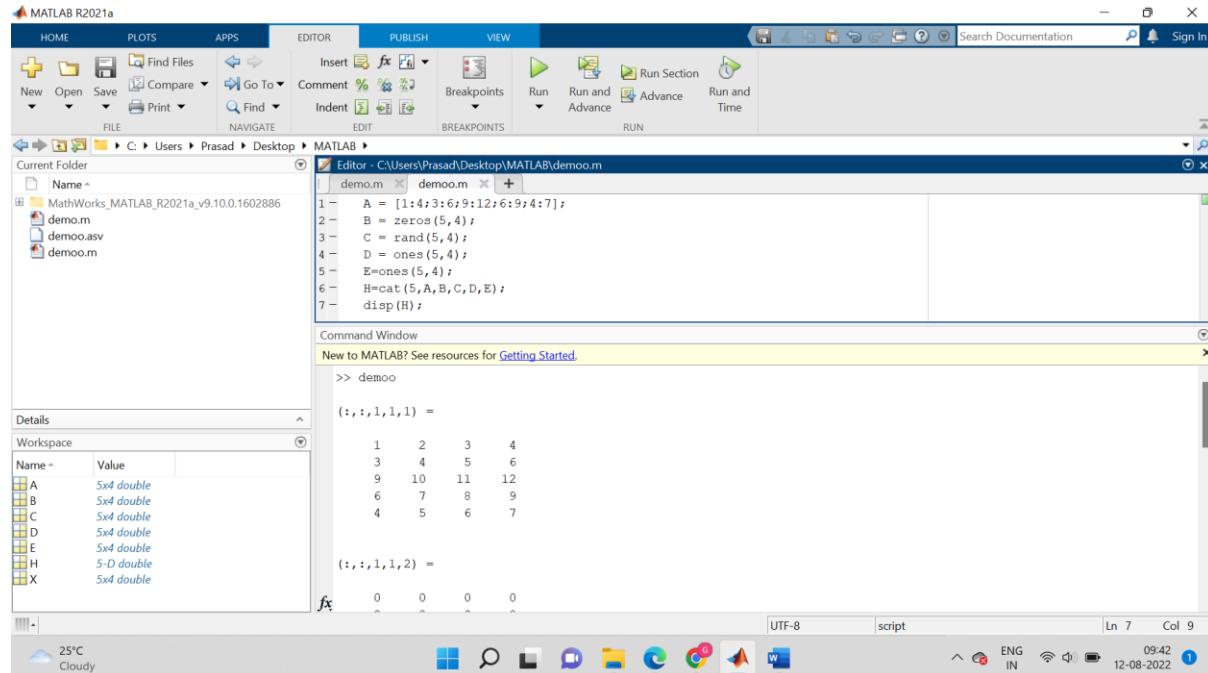


3. Demonstrate the use of cat () the creation of a multidimensional array of 5X4 with 5 dimensions.
 Use “ : range operator” to create one array, use “[] concatenation operator” to create another array, use appropriate special array functions.

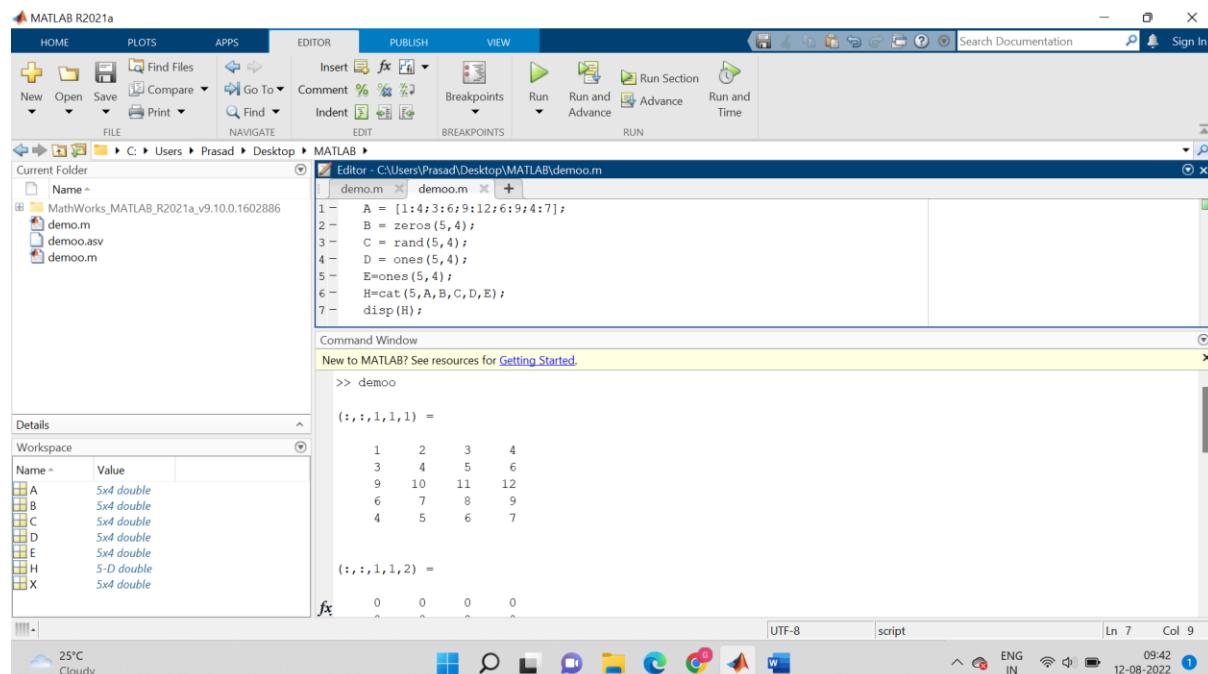
Explanation:

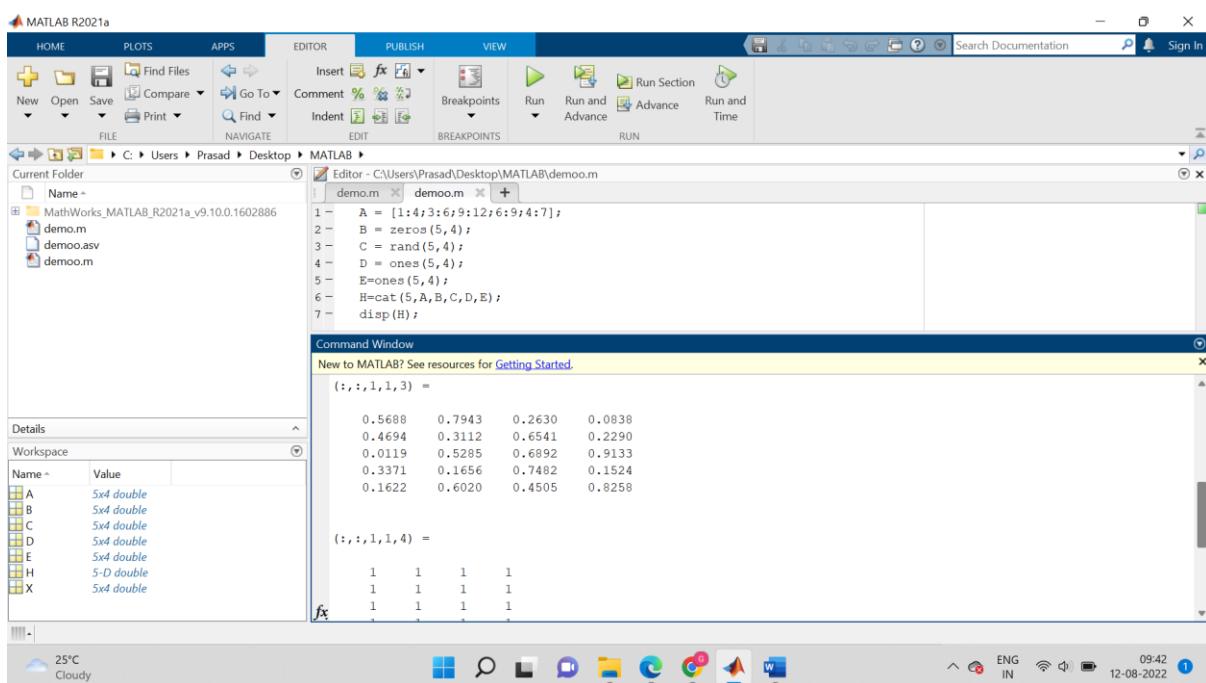
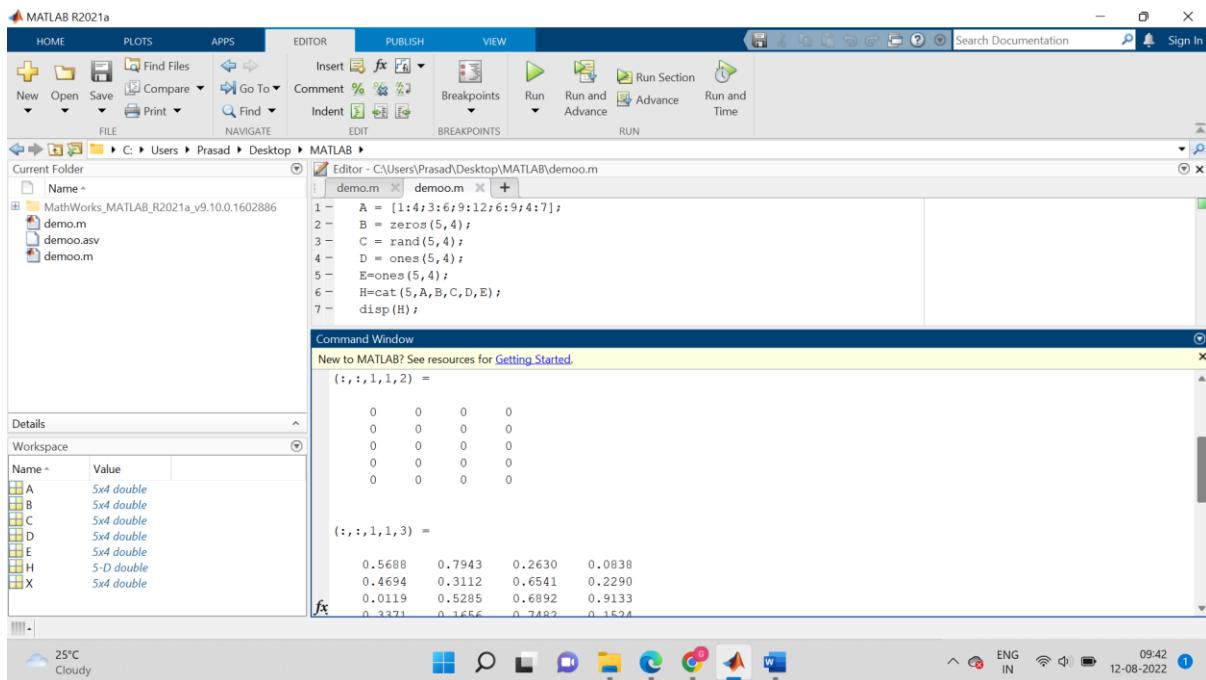
So, I have created 5 arrays of same dimensions. Use cat() function to concatenate them in multidimensional array.

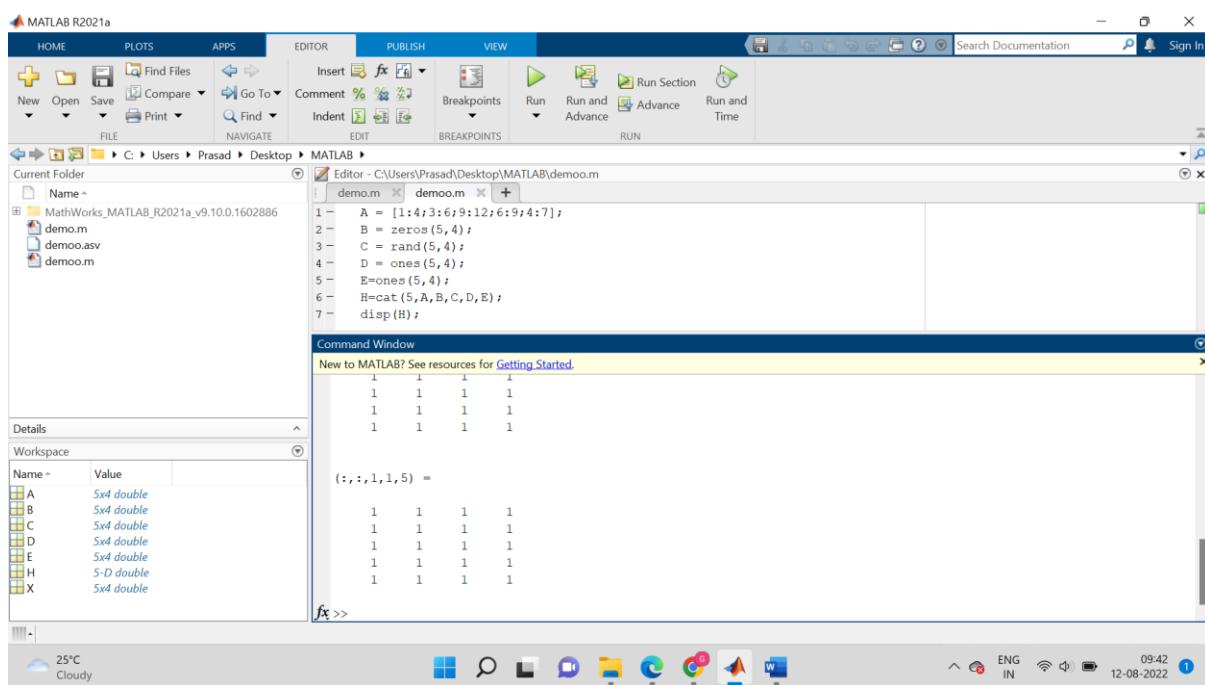
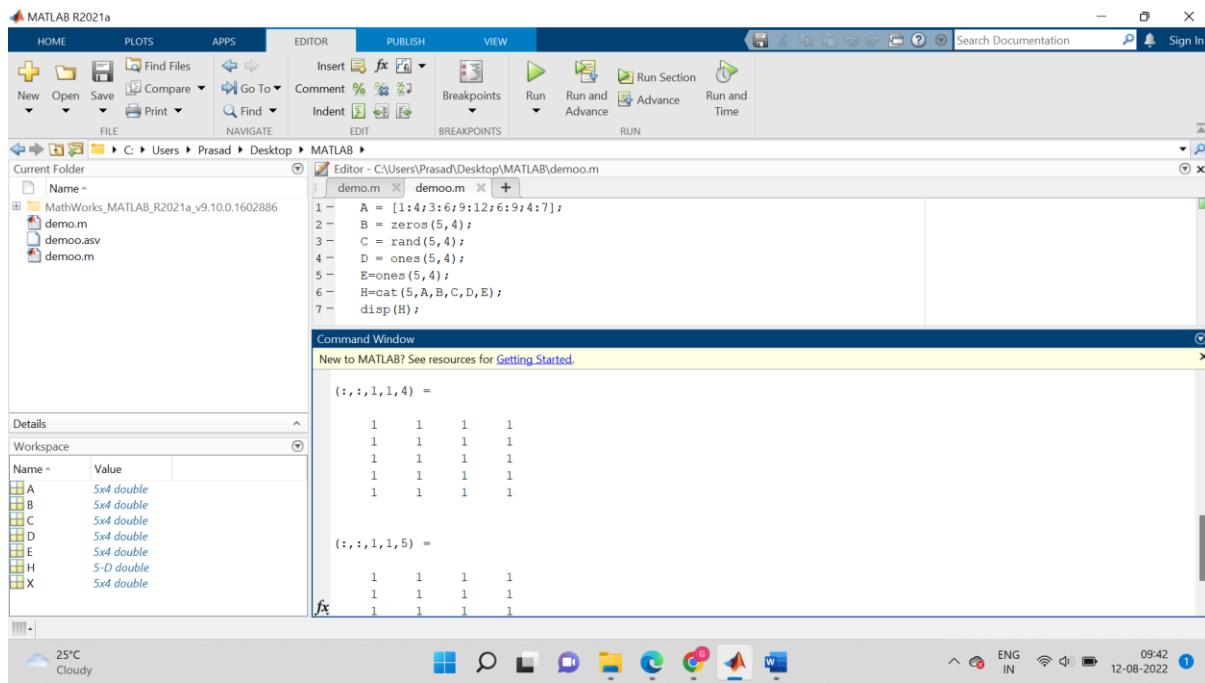
Code:



Output:

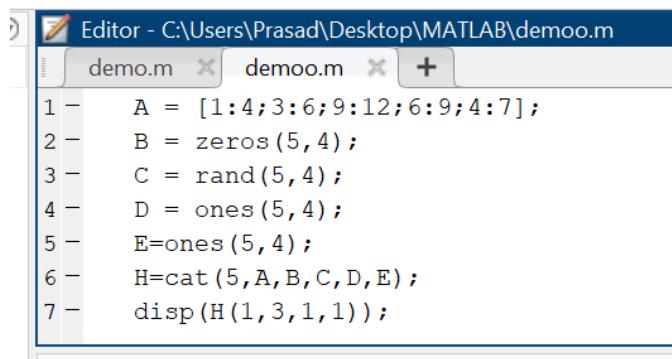






4. From the above array access the following elements:
- Any one element of a specific position in any dimension of your choice.

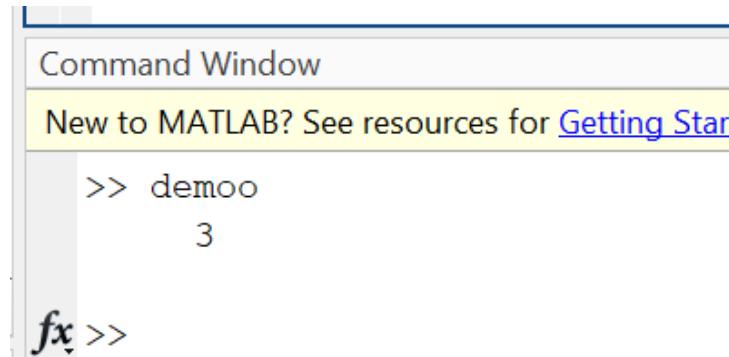
Code:



The screenshot shows the MATLAB Editor window with the file 'demoo.m' open. The code in the editor is as follows:

```
1 - A = [1:4;3:6;9:12;6:9;4:7];
2 - B = zeros(5,4);
3 - C = rand(5,4);
4 - D = ones(5,4);
5 - E=ones(5,4);
6 - H=cat(5,A,B,C,D,E);
7 - disp(H(1,3,1,1));
```

Output:



The screenshot shows the MATLAB Command Window. The user has entered the command `>> demoo`. The output is displayed in blue text, showing the value `3`.

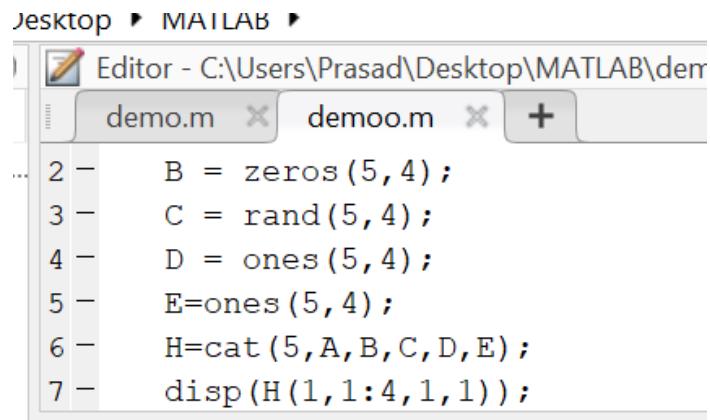
Command Window

New to MATLAB? See resources for [Getting Started](#)

```
>> demoo
3
```

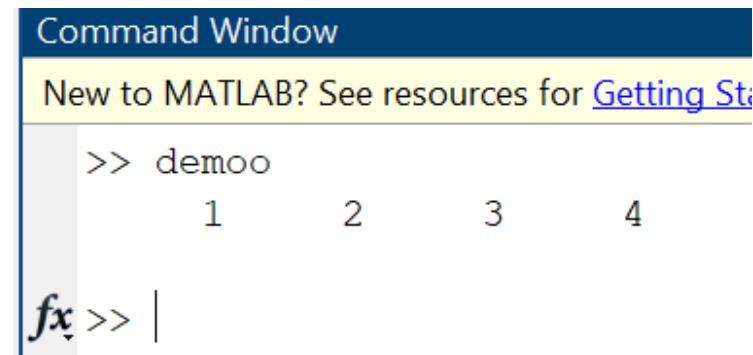
ii. Range of any four elements in a row in any dimension of your choice.

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\der
demo.m    demoo.m    +
2 - B = zeros(5, 4);
3 - C = rand(5, 4);
4 - D = ones(5, 4);
5 - E=ones(5, 4);
6 - H=cat(5,A,B,C,D,E);
7 - disp(H(1,1:4,1,1));
```

Output:



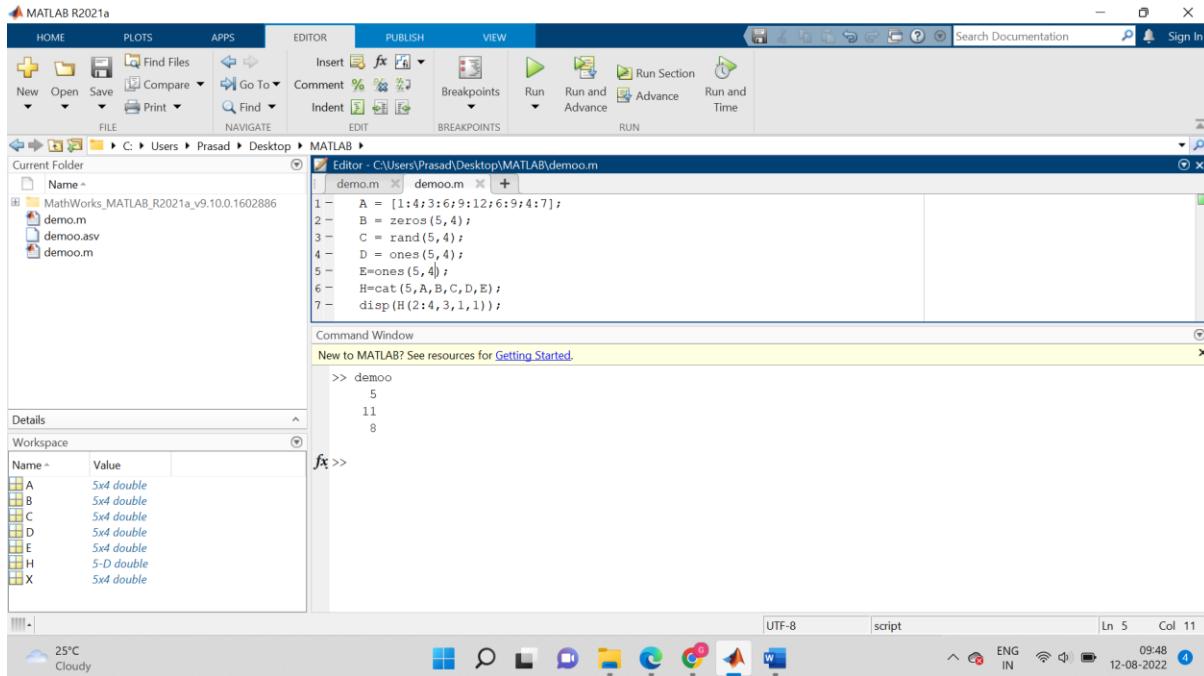
Command Window

New to MATLAB? See resources for [Getting Started](#).

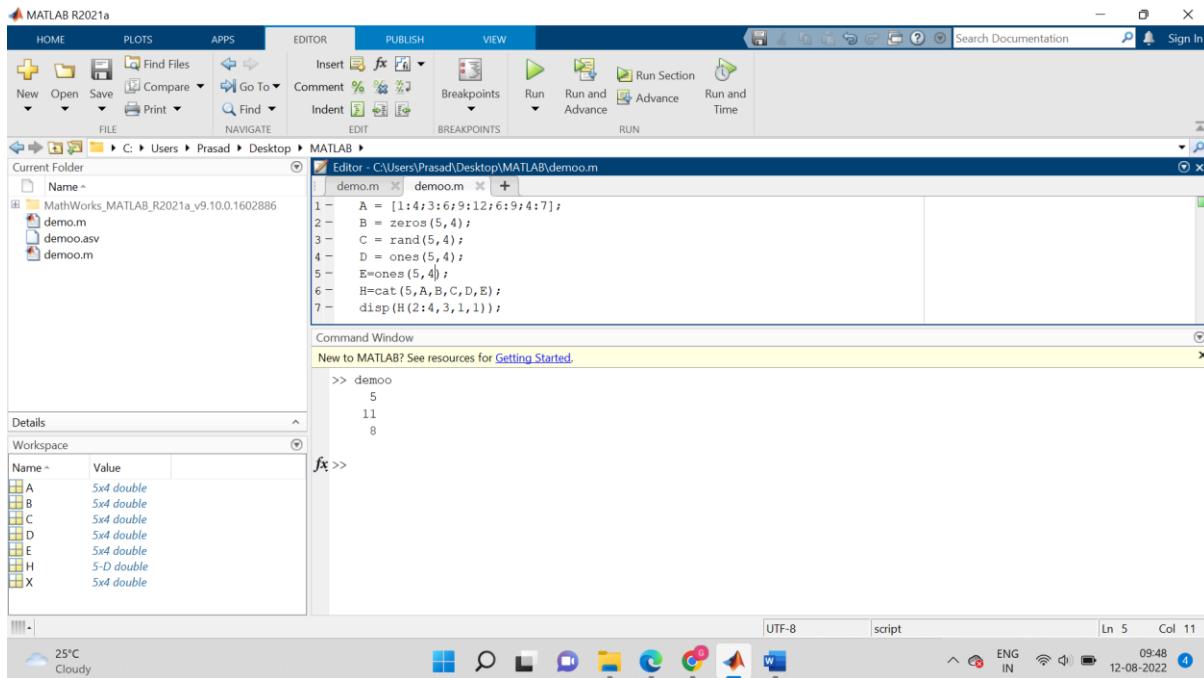
```
>> demoo
1      2      3      4
fx >> |
```

iii. Range of any three elements in a column in any dimension of your choice.

Code:

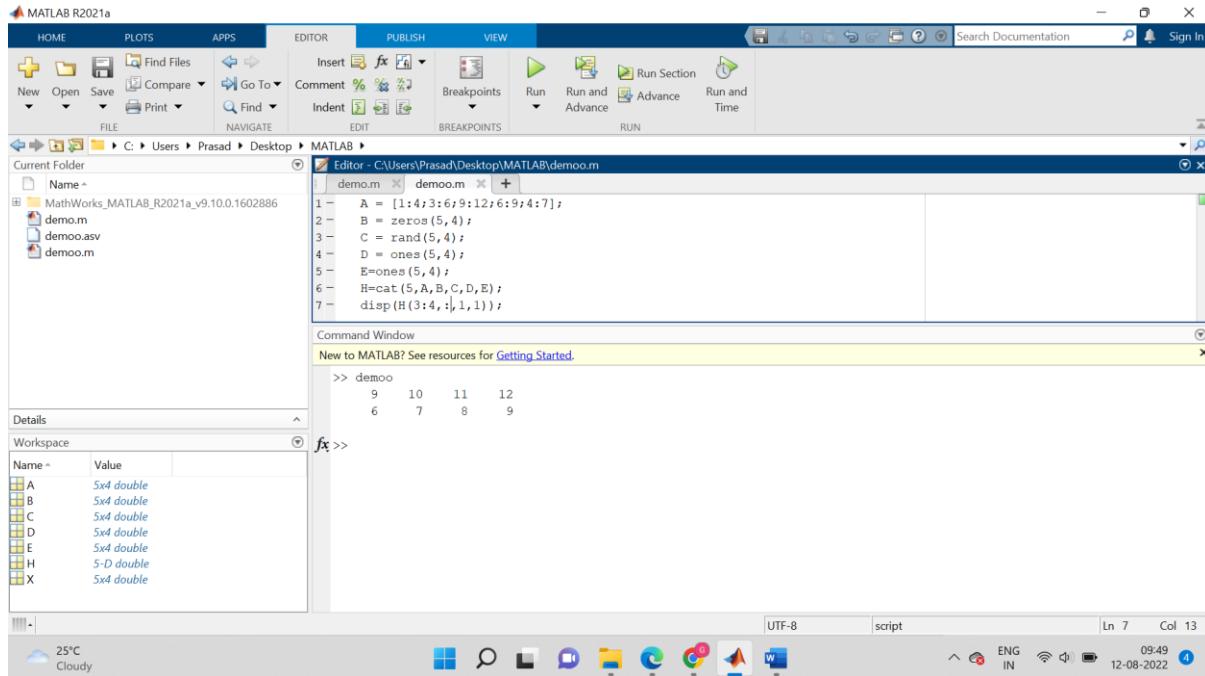


Output:

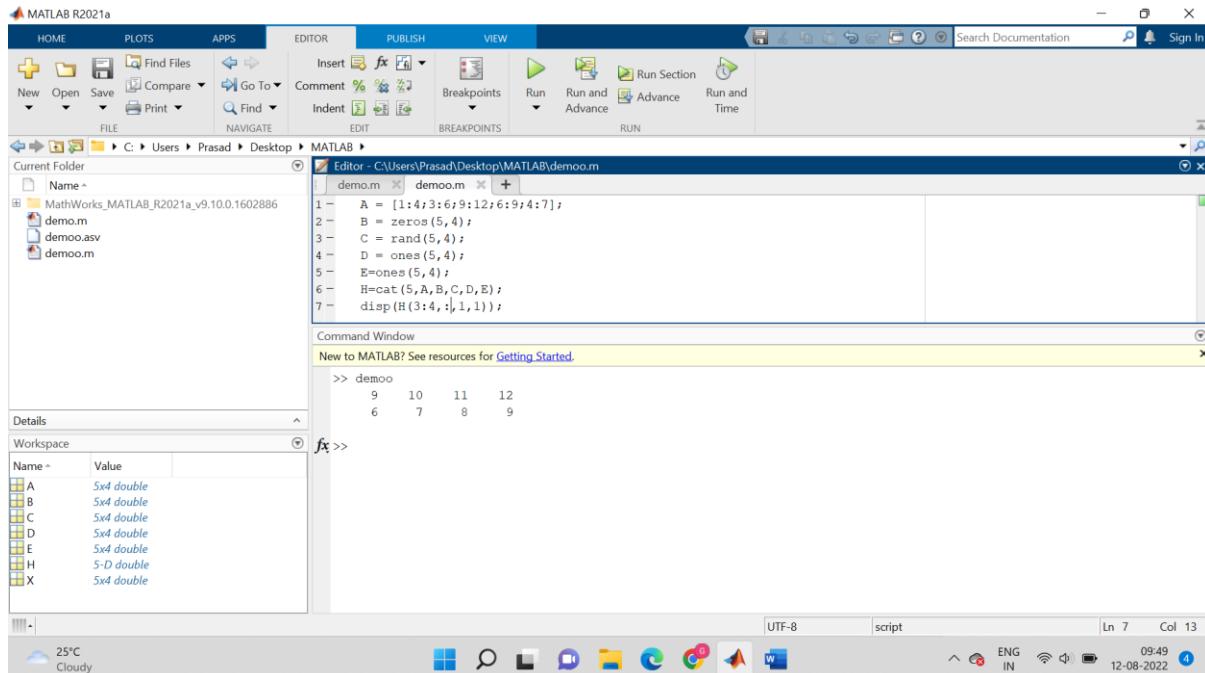


iv. All the elements of two rows in any dimension of your choice.

Code:

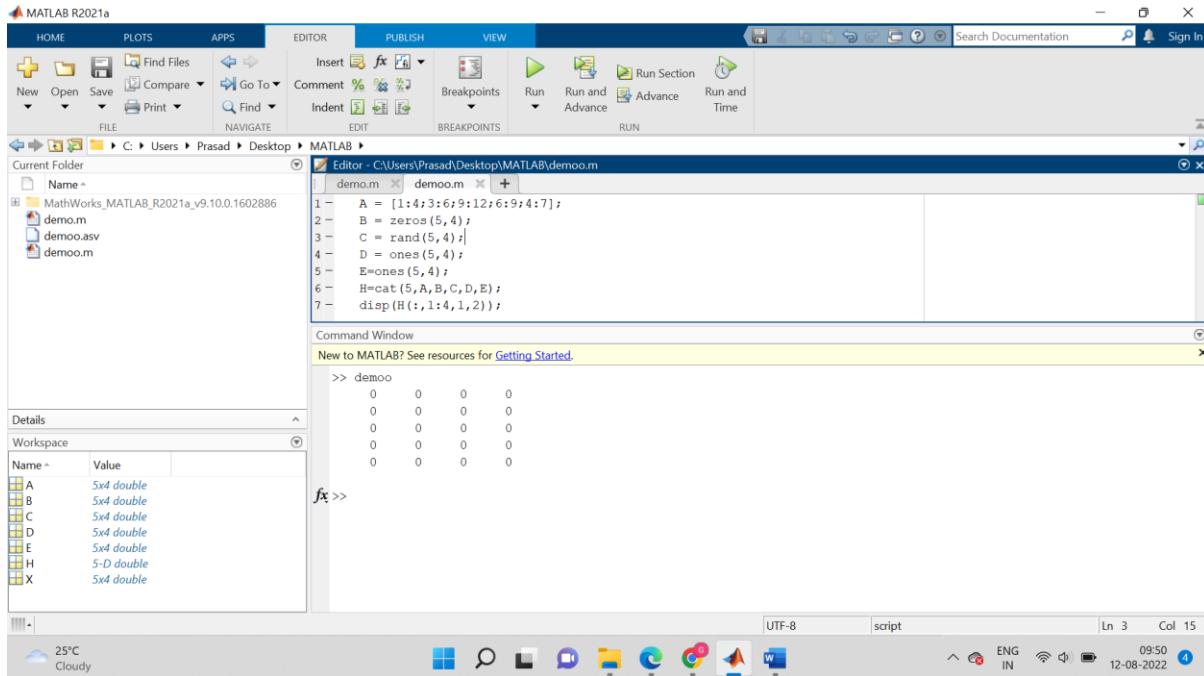


Output:

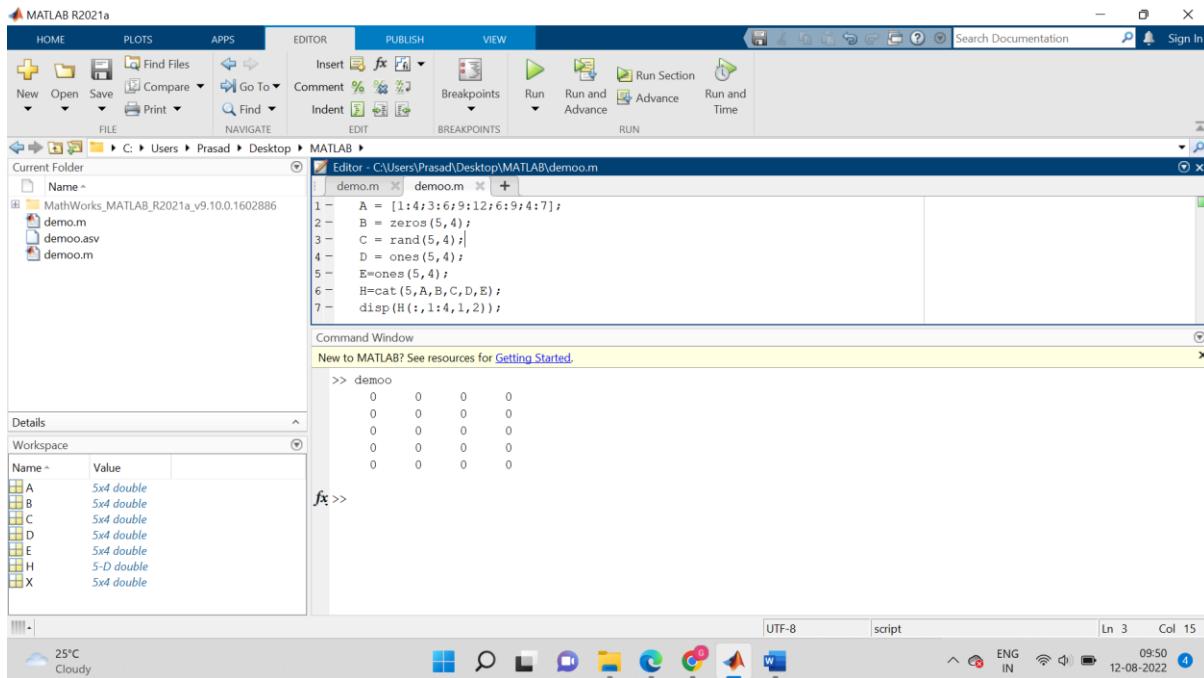


v. All the elements of any 4 columns in any dimension of your choice.

Code:



Output:

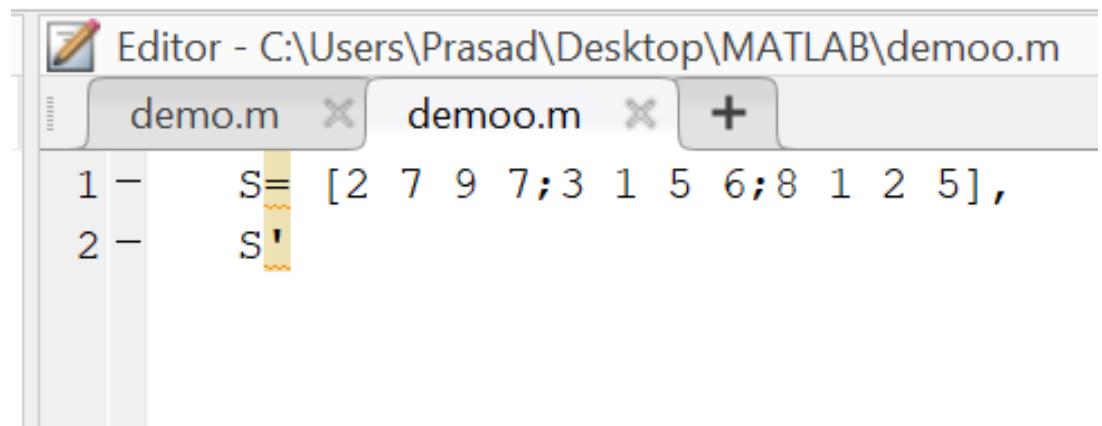


Colon Notation:

1. If $S = [2 \ 7 \ 9 \ 7; 3 \ 1 \ 5 \ 6; 8 \ 1 \ 2 \ 5]$, do the following actions
 - a. S'

Explanation:**Code:**

MATLAB ▶

**Command Window****Output:**

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> demoo
S =
    2     7     9     7
    3     1     5     6
    8     1     2     5

ans =
    2     3     8
    7     1     1
    9     5     2
    7     6     5
```

fx >> |

b. $S(:,[1\ 4])$

Code:

The screenshot shows the MATLAB Editor window titled "Editor - C:\Users\Admin\Documents\MATLAB\demo.m". It contains two lines of code: line 1 defines a matrix S with elements [2 7 9 7; 3 1 5 6; 8 1 2 5]; line 2 shows the command S(:, [1 4]). The second line is highlighted with a yellow selection bar.

```
demo.m
1 - S= [2 7 9 7;3 1 5 6;8 1 2 5];
2 - S(:, [1 4])
```

Output:

The screenshot shows the MATLAB Command Window. The user types ">> demo" and presses Enter. The output is: ans =
2 7
3 6
8 5
The prompt ">>" appears again at the bottom.

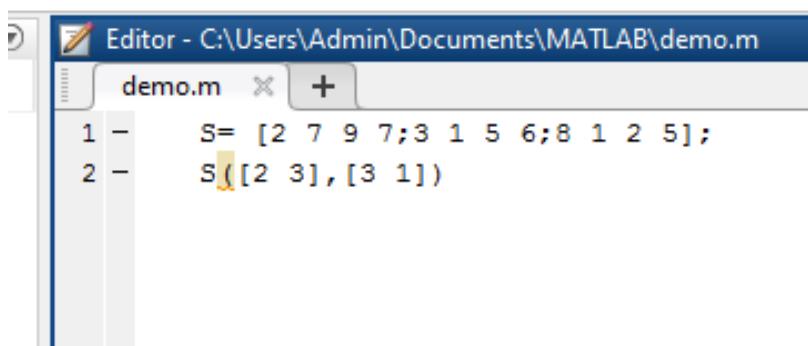
```
Command Window
New to MATLAB? See resources for Getting Started.
>> demo

ans =
    2     7
    3     6
    8     5

>>
```

c. $S([2 \ 3],[3 \ 1])$

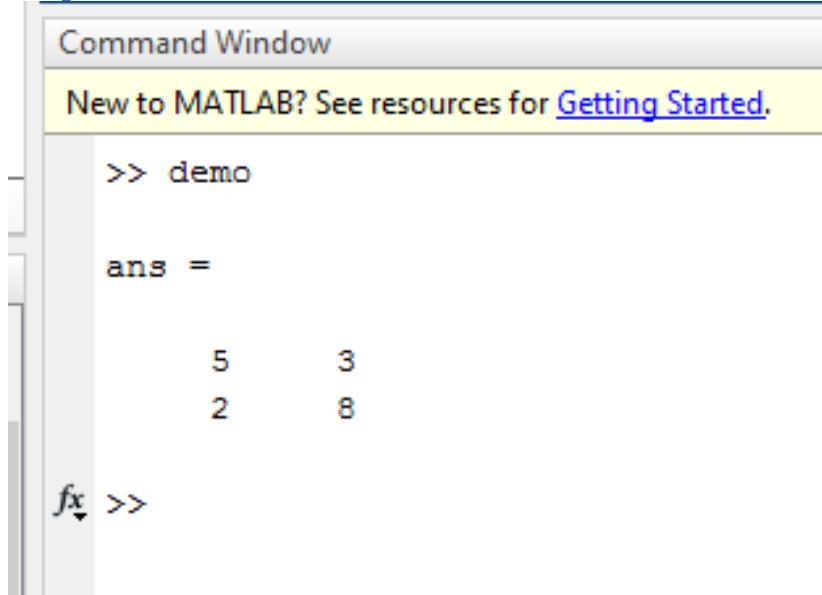
Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code in the editor is:

```
1 - S= [2 7 9 7;3 1 5 6;8 1 2 5];
2 - S([2 3], [3 1])
```

Output:

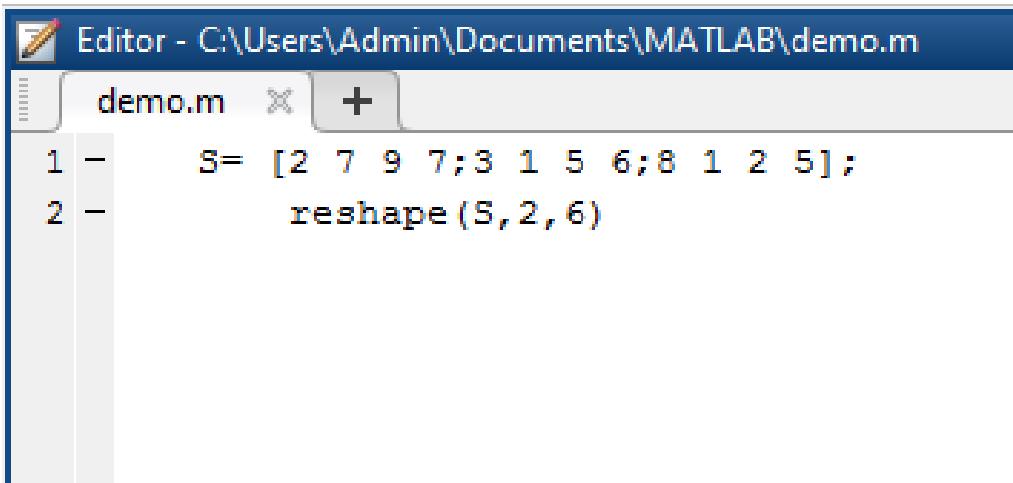


The screenshot shows the MATLAB Command Window. The user typed '`>> demo`' and pressed Enter. The output was:

```
ans =
      5     3
      2     8
```

d. `reshape(S,2,6)`

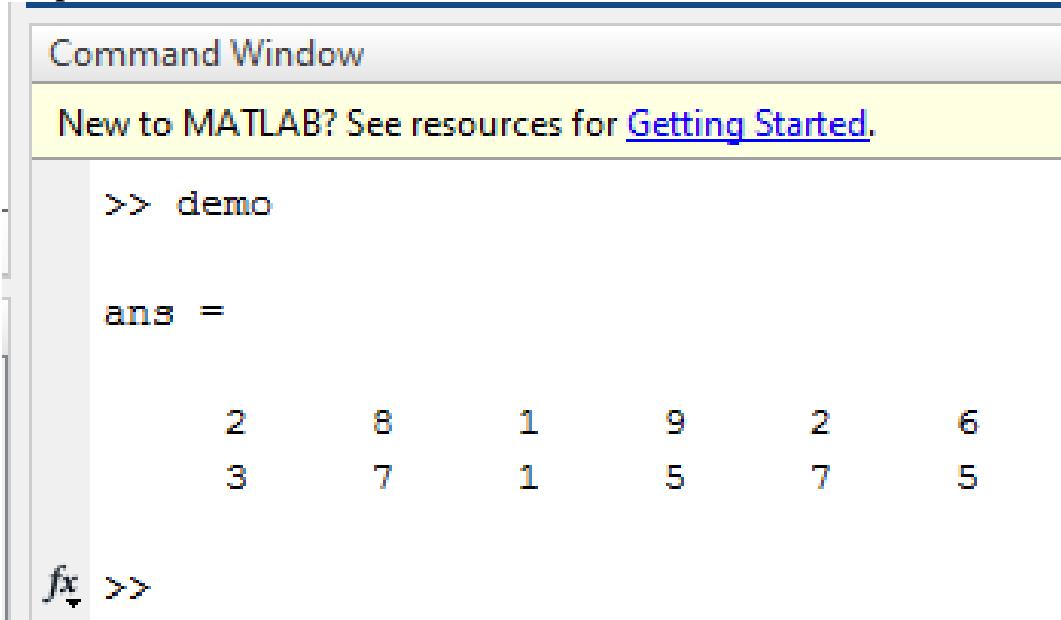
Code:



The screenshot shows the MATLAB Editor window with a single file named "demo.m". The code in the editor is as follows:

```
1 - S= [2 7 9 7;3 1 5 6;8 1 2 5];
2 - reshape (S, 2, 6)
```

Output:



The screenshot shows the MATLAB Command Window. It displays the output of running the "demo" script. The output is:

```
>> demo

ans =

    2     8     1     9     2     6
    3     7     1     5     7     5
```

e. $S(:)$

Code:

The screenshot shows the MATLAB Editor window titled "Editor - C:\Users\Admin\Documents\MATLAB\demo.m". It contains two lines of code: line 1 defines a matrix S with elements 2, 7, 9, 7, 3, 1, 5, 6, 8, 1, 2, 5; line 2 attempts to extract all rows of S using the command $S(:)$. The colon in this command is highlighted with a yellow selection bar.

```
1 - S= [2 7 9 7;3 1 5 6;8 1 2 5];
2 - S(:)|
```

Output:

The screenshot shows the MATLAB Command Window. It starts with a welcome message: "New to MATLAB? See resources for [Getting Started](#)". Then it runs the command `>> demo`, which outputs the matrix S row by row. The matrix S is defined as:

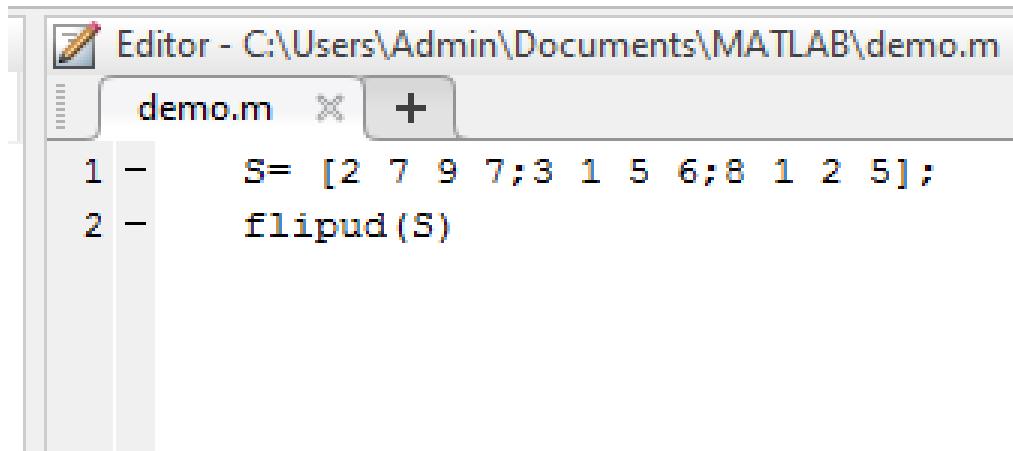
$$S = \begin{bmatrix} 2 & 7 & 9 & 7 \\ 3 & 1 & 5 & 6 \\ 8 & 1 & 2 & 5 \end{bmatrix}$$

The output in the Command Window is:

```
>> demo
ans =
2
3
8
7
1
1
9
5
2
7
6
5
```

f. `flipud(S)`

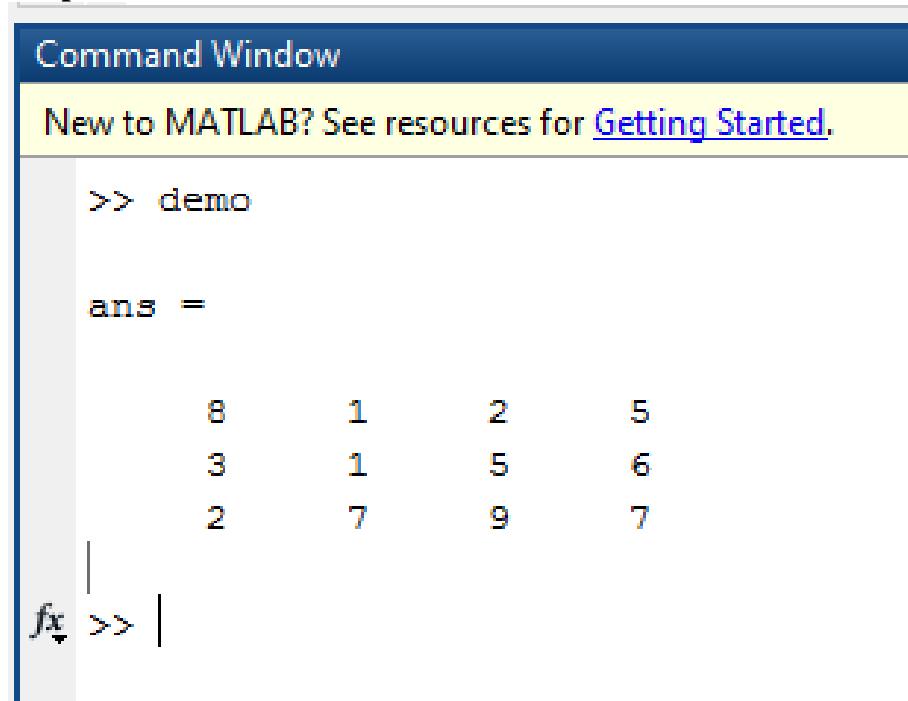
Code:



The screenshot shows the MATLAB Editor window with a single file named "demo.m". The code inside the file is:

```
1 - S= [2 7 9 7;3 1 5 6;8 1 2 5];
2 - flipud(S)
```

Output:



The screenshot shows the MATLAB Command Window. At the top, it says "Command Window" and "New to MATLAB? See resources for [Getting Started](#)". The command entered was ">> demo". The output displayed is:

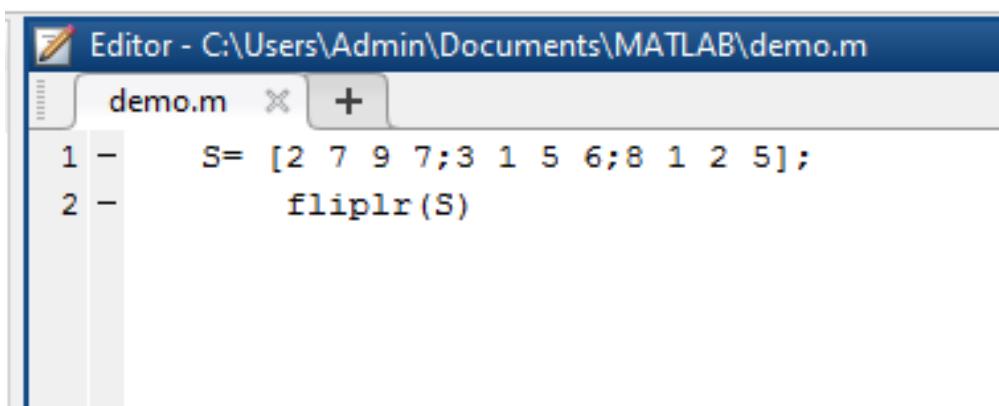
```
>> demo

ans =

    8     1     2     5
    3     1     5     6
    2     7     9     7
```

g. `fliplr(S)`

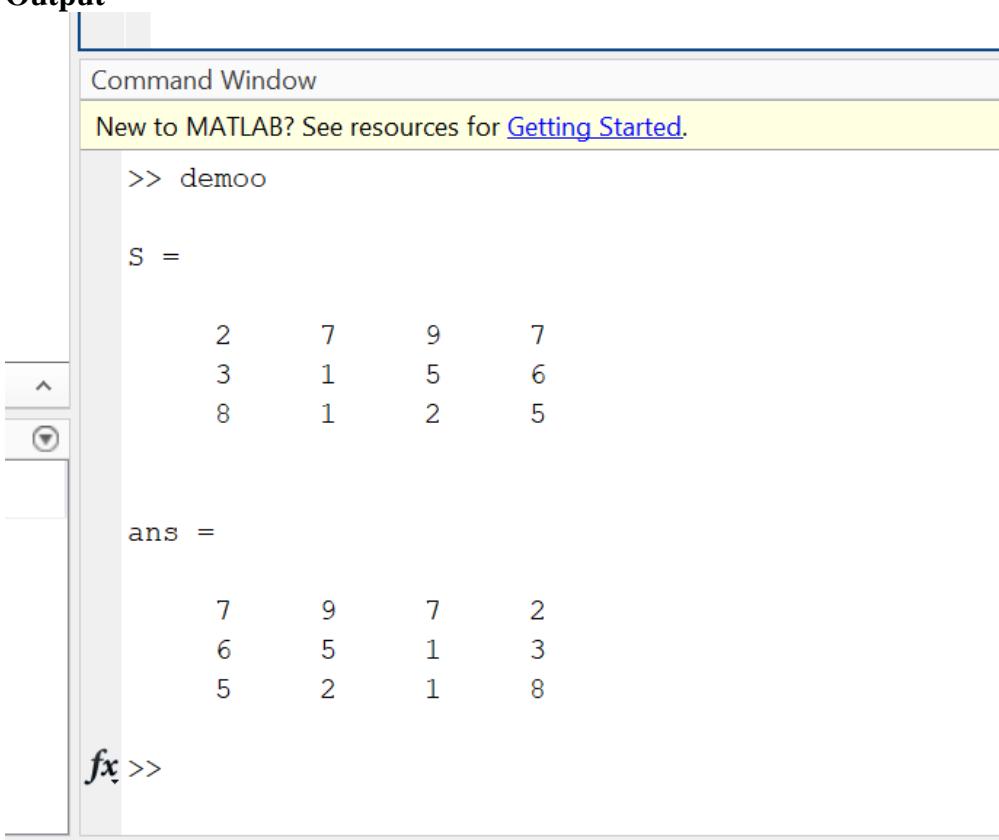
Code:



The screenshot shows the MATLAB Editor window with a single file named "demo.m". The code in the editor is:

```
1 - S= [2 7 9 7;3 1 5 6;8 1 2 5];
2 - fliplr(S)
```

Output



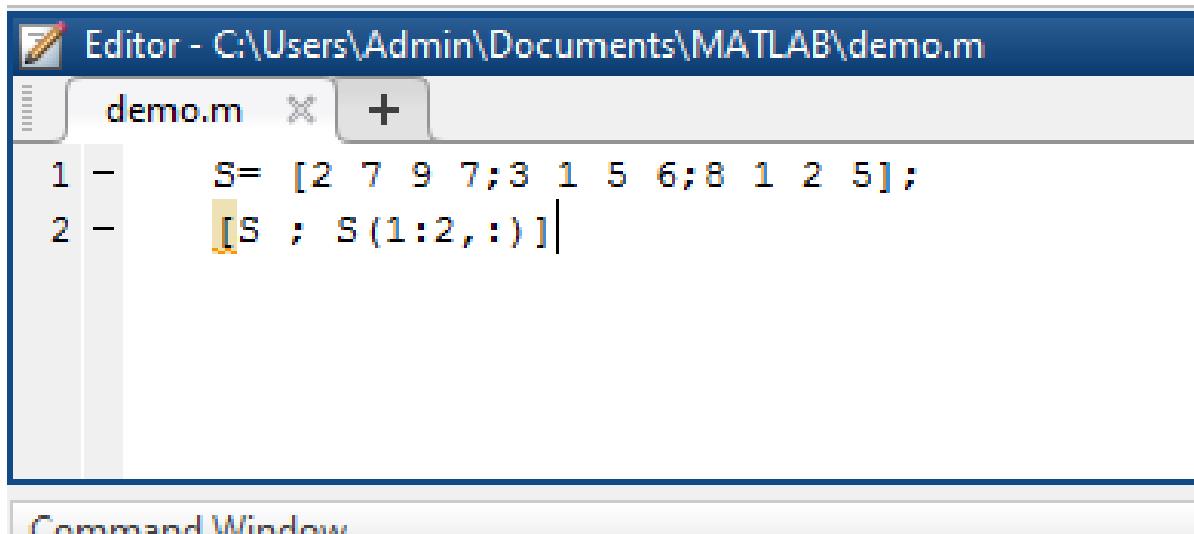
The screenshot shows the MATLAB Command Window. The user has run the script "demo.m". The output shows the original matrix S and its transpose ans.

```
>> demo.m
S =
    2     7     9     7
    3     1     5     6
    8     1     2     5

ans =
    7     9     7     2
    6     5     1     3
    5     2     1     8
```

h. $[S ; S(1:2,:)]$

Code:

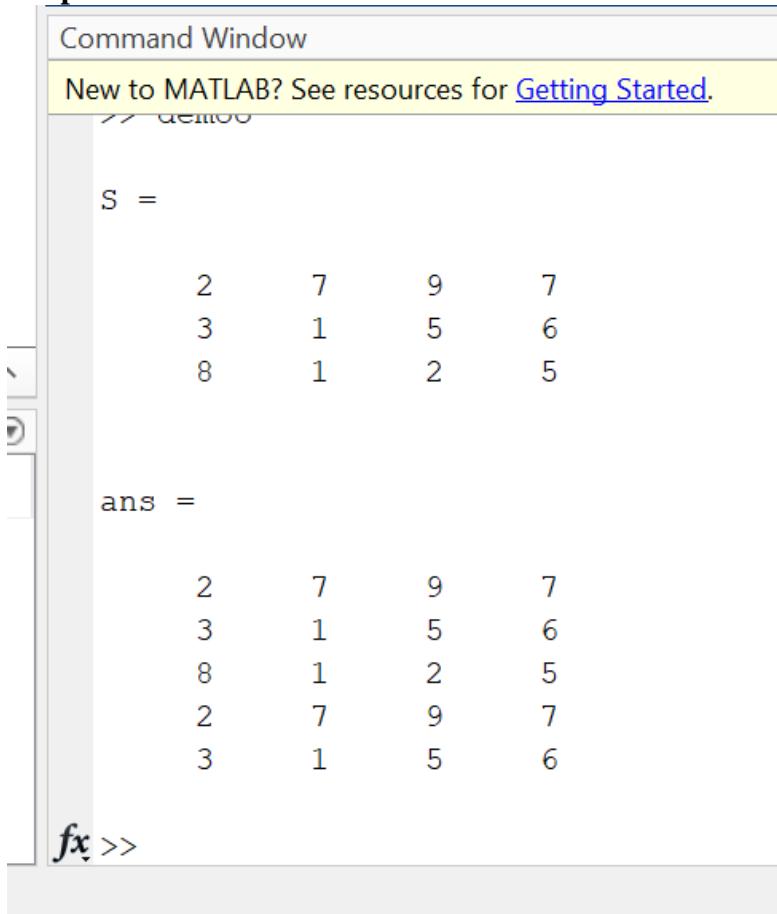


The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code in the editor is:

```
1 - S= [2 7 9 7;3 1 5 6;8 1 2 5];
2 - [S ; S(1:2, :)]
```

Command Window

Output:



The screenshot shows the MATLAB Command Window. The output is:

```
New to MATLAB? See resources for Getting Started.
// demo

S =
    2     7     9     7
    3     1     5     6
    8     1     2     5

ans =
    2     7     9     7
    3     1     5     6
    8     1     2     5
    2     7     9     7
    3     1     5     6
```

i. `sum(S)`

Code:

The screenshot shows the MATLAB Editor window with the file `demo.m` open. The code in the editor is:

```
1 - S= [2 7 9 7;3 1 5 6;8 1 2 5];
2 - sum (S)
```

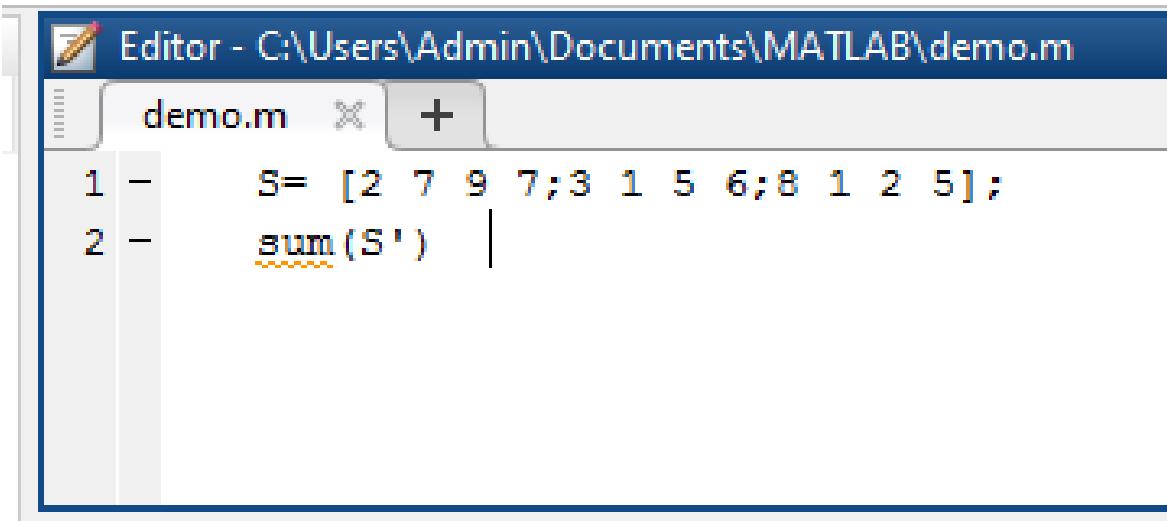
Output

The screenshot shows the MATLAB Command Window. The command `>> demo` was entered, and the output is:

```
ans =
    13     9     16     18
```

j. $\text{sum}(S')$

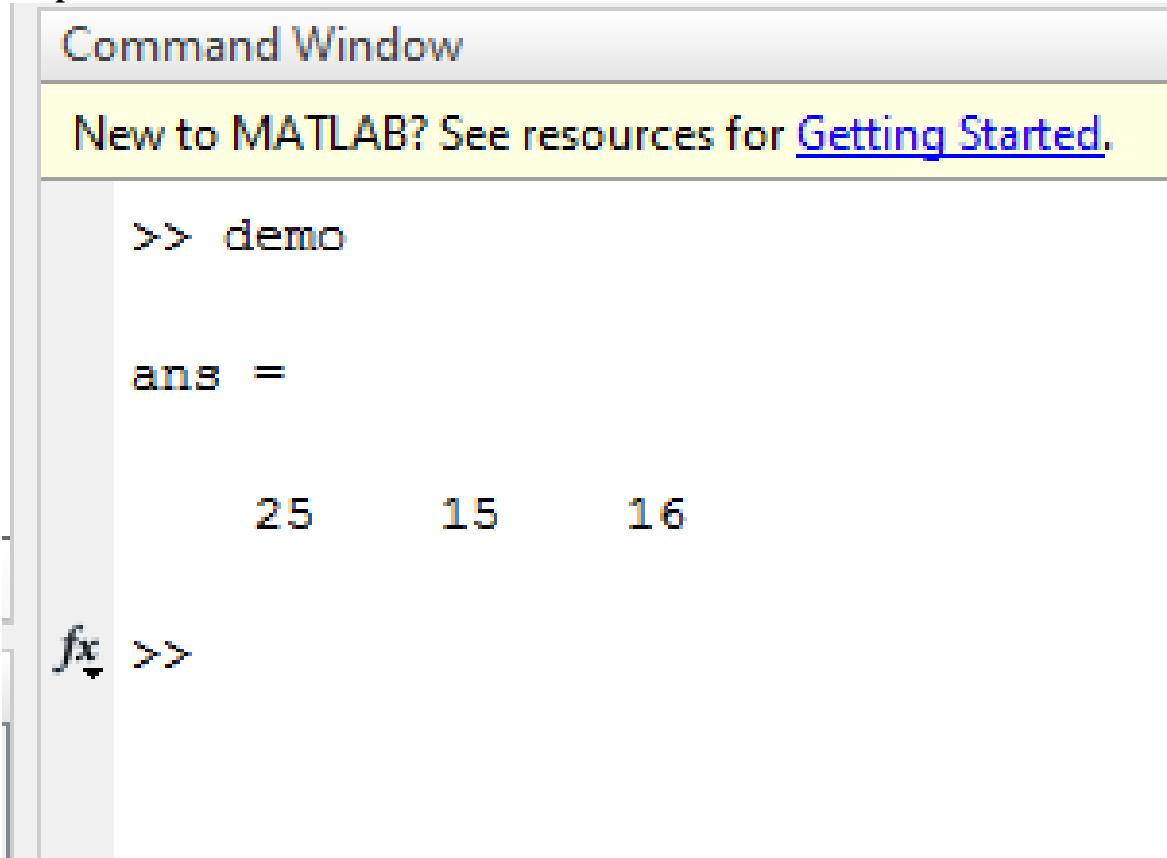
Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code in the editor is:

```
S= [2 7 9 7;3 1 5 6;8 1 2 5];
sum(S')
```

Output

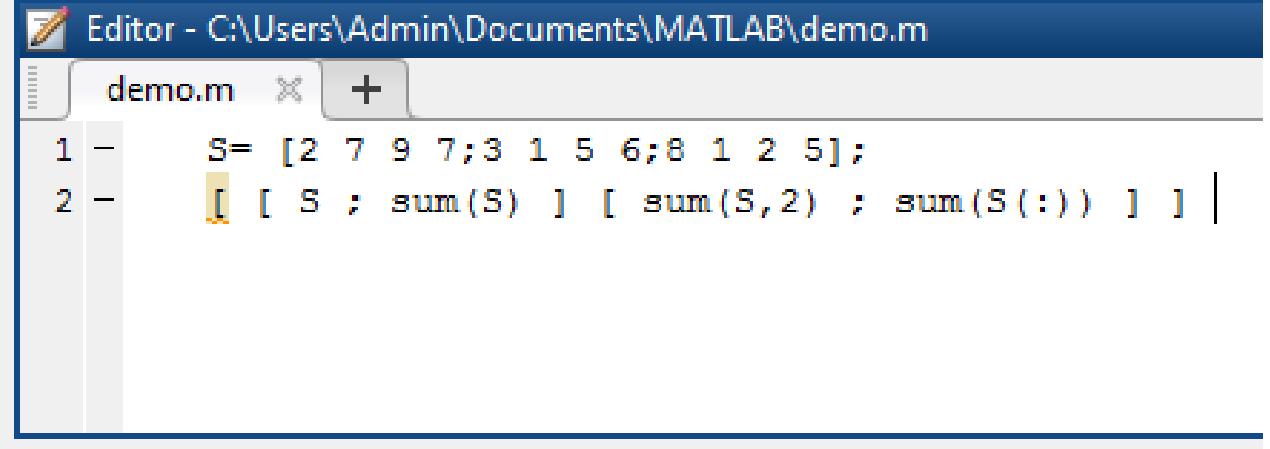


The screenshot shows the MATLAB Command Window. It displays the command `>> demo` and its output:

```
ans =
    25      15      16
```

k. $[[S; \text{sum}(S)][\text{sum}(S, 2); \text{sum}(S(:))]]$

Code:



The screenshot shows the MATLAB Editor window with the file name 'demo.m'. The code in the editor is:

```
1 - S= [2 7 9 7;3 1 5 6;8 1 2 5];
2 - [ [ S ; sum(S) ] [ sum(S,2) ; sum(S(:)) ] ]
```

Output

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> demo
```

```
ans =
```

2	7	9	7	25
3	1	5	6	15
8	1	2	5	16
13	9	16	18	56

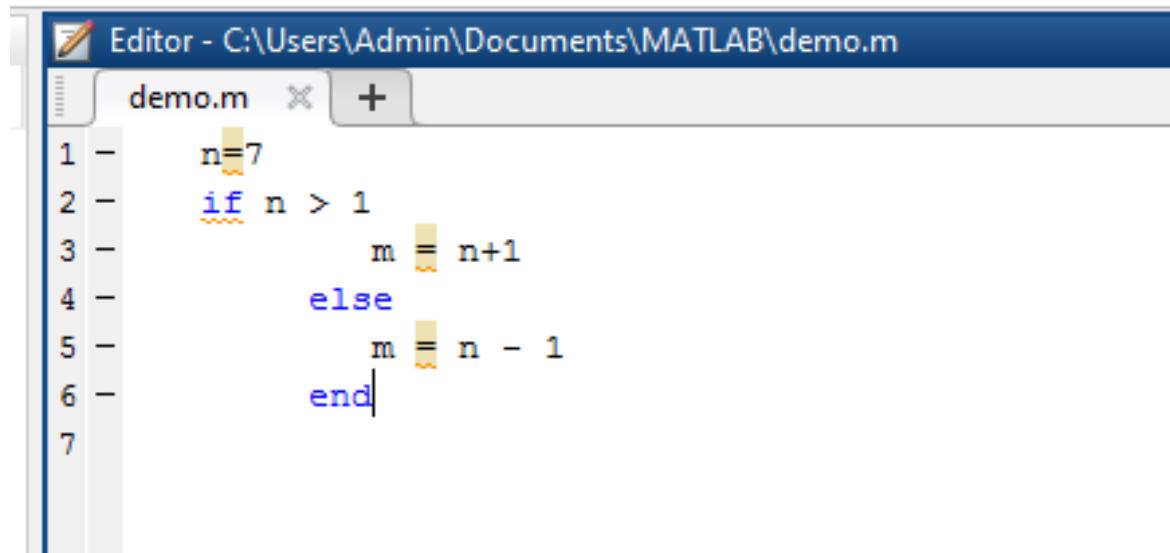
fx >>

```
1. if n > 1  
    m = n+1  
else  
    m = n - 1  
end
```

- a. n = 7 m = ?
- b. n = 0 m = ?
- c. n = -10 m = ?

a. n = 7 m = ?

Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code is as follows:

```
1 - n=7  
2 - if n > 1  
3 -     m = n+1  
4 - else  
5 -     m = n - 1  
6 - end  
7
```

Output:

New to MATLAB? See resources for [Getting Started](#).

```
>> demo
```

```
n =
```

```
7
```

```
m =
```

```
8
```

```
fx >>
```

b. $n = 0 \quad m = ?$

Code:

The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code is as follows:

```
1 - n=0
2 - if n > 1
3 -     m = n+1
4 - else
5 -     m = n - 1
6 - end
7
```

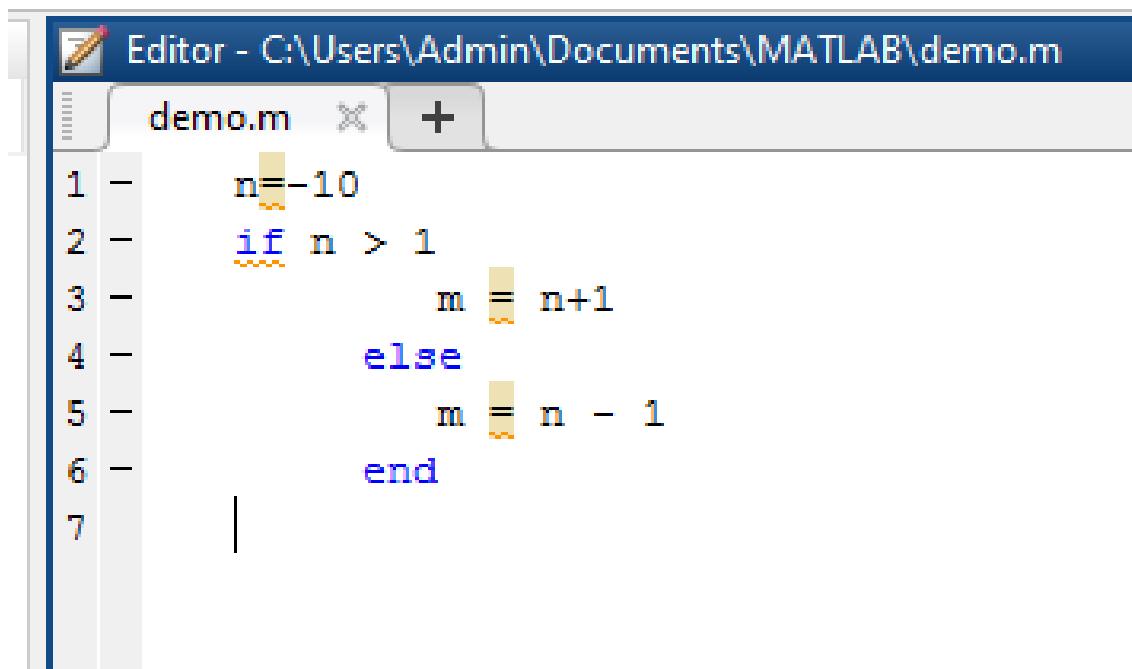
Output:

The screenshot shows the MATLAB Command Window. The user types '`>> demo`' and presses Enter. The output shows:

```
8
>> demo
n =
0
m =
-1
fx >>
```

c.n = -10 m = ?

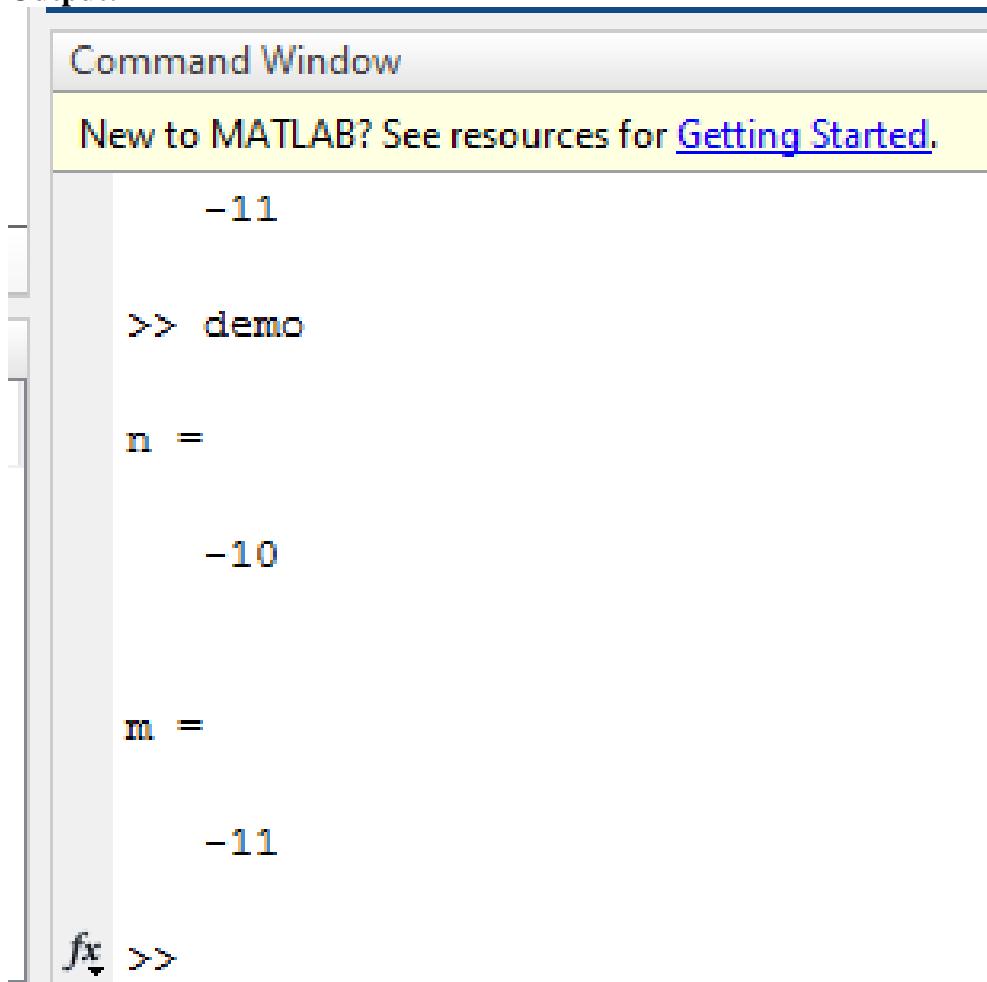
Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code is as follows:

```
1 - n = -10
2 - if n > 1
3 -     m = n+1
4 - else
5 -     m = n - 1
6 - end
7 -
```

Output:



The screenshot shows the MATLAB Command Window output. The user runs the command `>> demo`. The window displays the following text:

```
-11

>> demo

n =
-10

m =
-11

fx >>
```

```

2. if z < 5
   w = 2*z
elseif z < 10
   w = 9 - z
elseif z < 100
   w = sqrt(z)
else
   w = z
end

```

a. z = 1 w = ?

Code:

The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code is as follows:

```

1 z=1
2 if z < 5
3   w = 2*z
4 elseif z < 10
5   w = 9 - z
6 elseif z < 100
7   w = sqrt(z)
8 else
9   w = z
10 end

```

Output:

The screenshot shows the MATLAB Command Window. The user types 'demo' and presses enter. The output shows the values of variables z and w.

```

>> demo

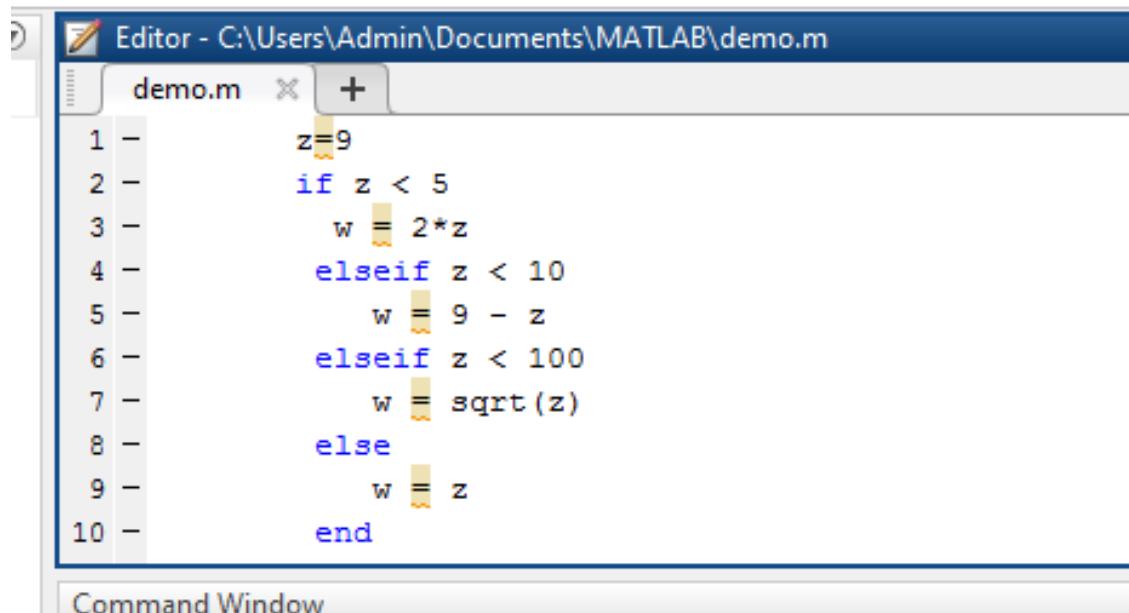
z =
1

w =
2

```

b. . z = 9 w = ?

Code:

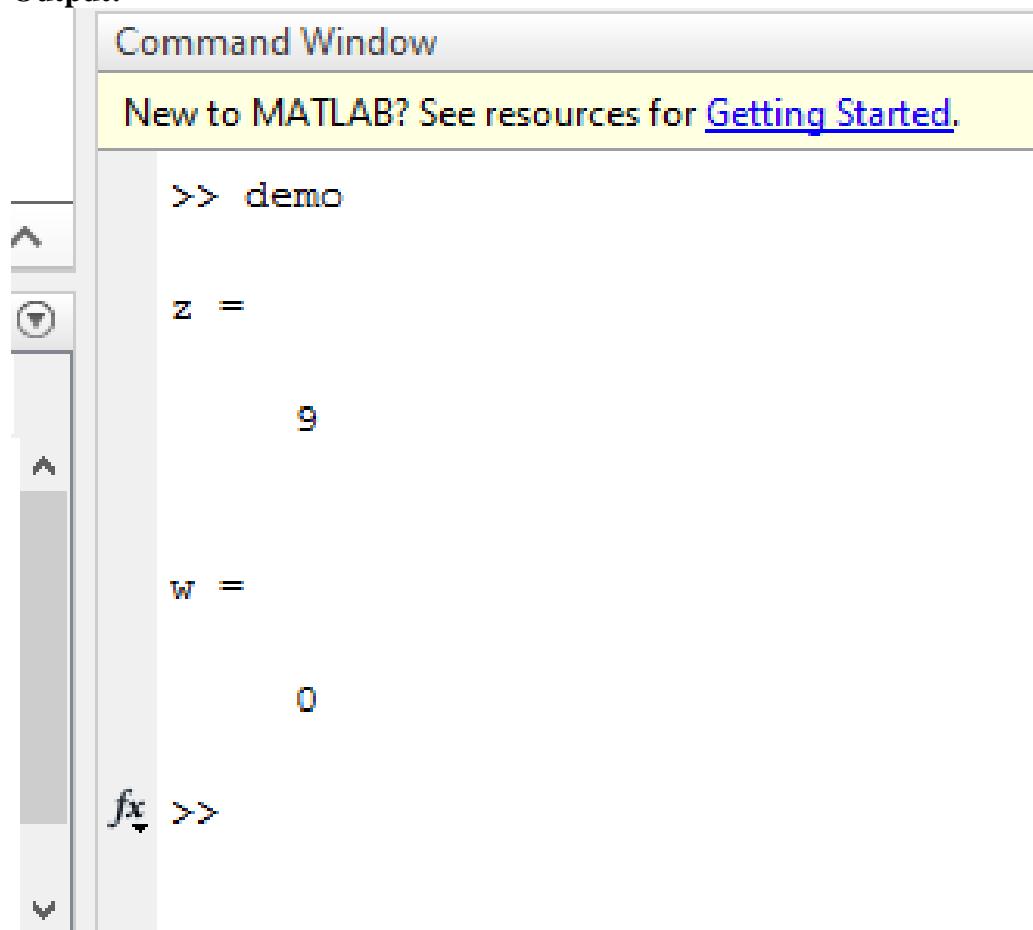


The screenshot shows the MATLAB Editor window with a file named 'demo.m'. The code inside the file is as follows:

```
1 -         z=9
2 -     if z < 5
3 -         w = 2*z
4 -     elseif z < 10
5 -         w = 9 - z
6 -     elseif z < 100
7 -         w = sqrt(z)
8 -     else
9 -         w = z
10 -    end
```

Below the editor is a Command Window tab.

Output:



The screenshot shows the MATLAB Command Window displaying the output of running the 'demo' script. The output is:

```
>> demo

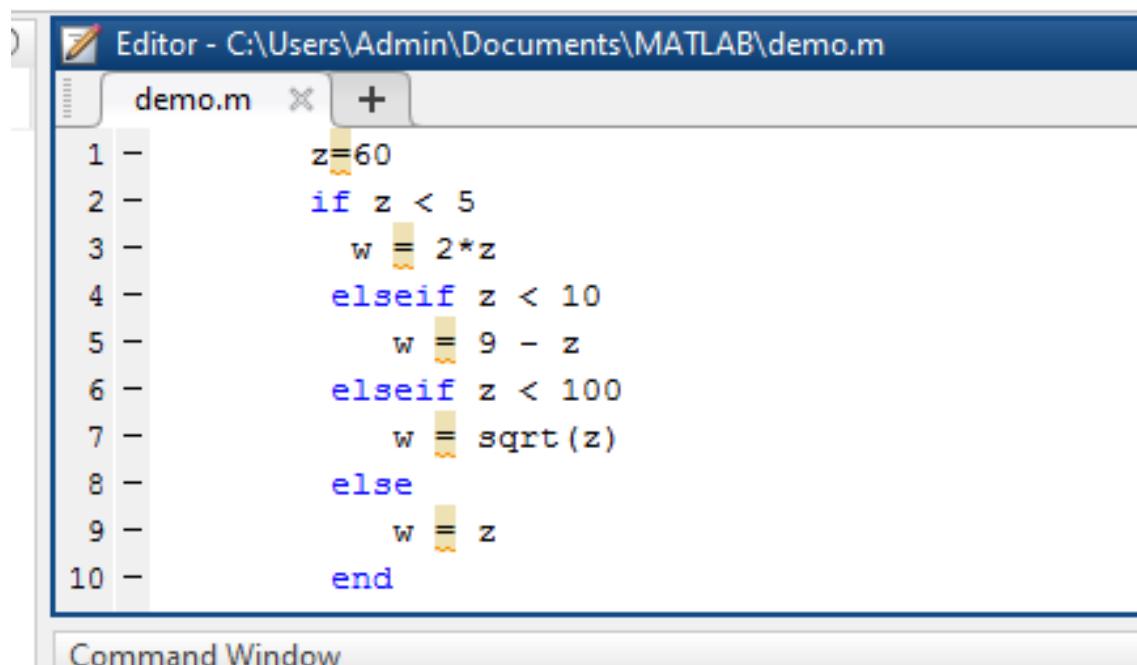
z =
9

w =
0

fx >>
```

c. $z = 60$ $w = ?$

Code:

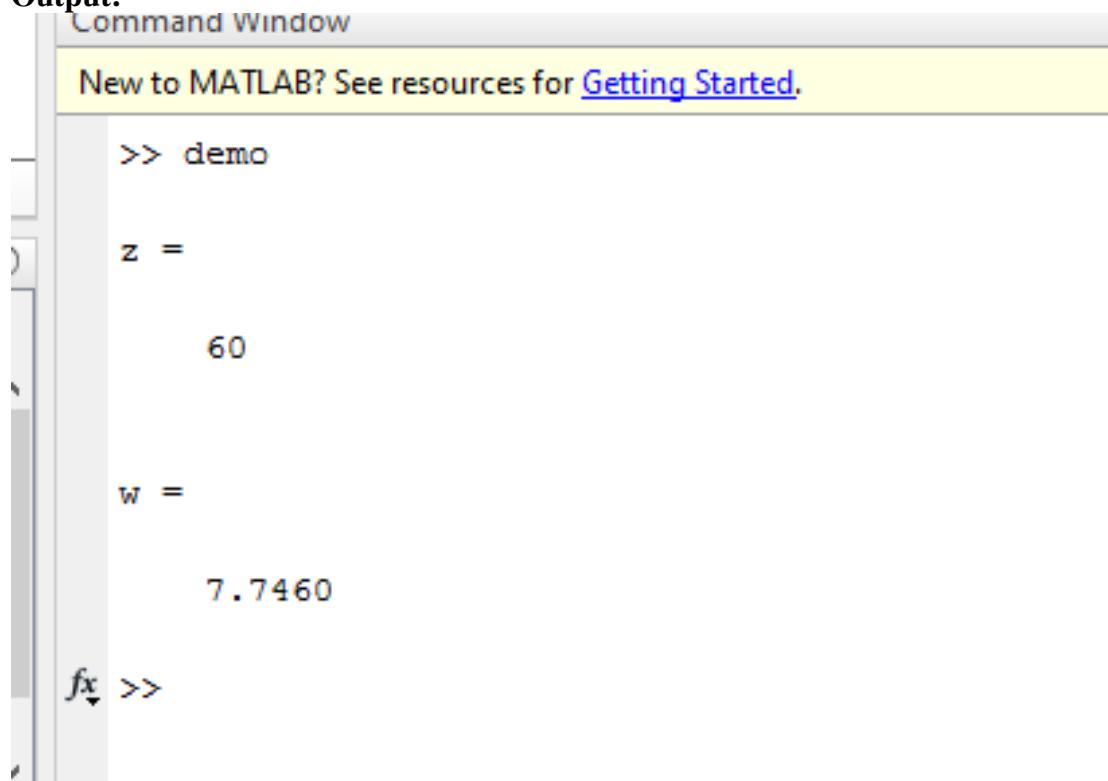


The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code defines a variable z and uses an if-elseif-else structure to calculate w based on the value of z . The code is as follows:

```
1 - z=60
2 - if z < 5
3 -     w = 2*z
4 - elseif z < 10
5 -     w = 9 - z
6 - elseif z < 100
7 -     w = sqrt(z)
8 - else
9 -     w = z
10 - end
```

Command Window

Output:



The screenshot shows the MATLAB Command Window. After running the script 'demo', it displays the values of z and w .

```
>> demo

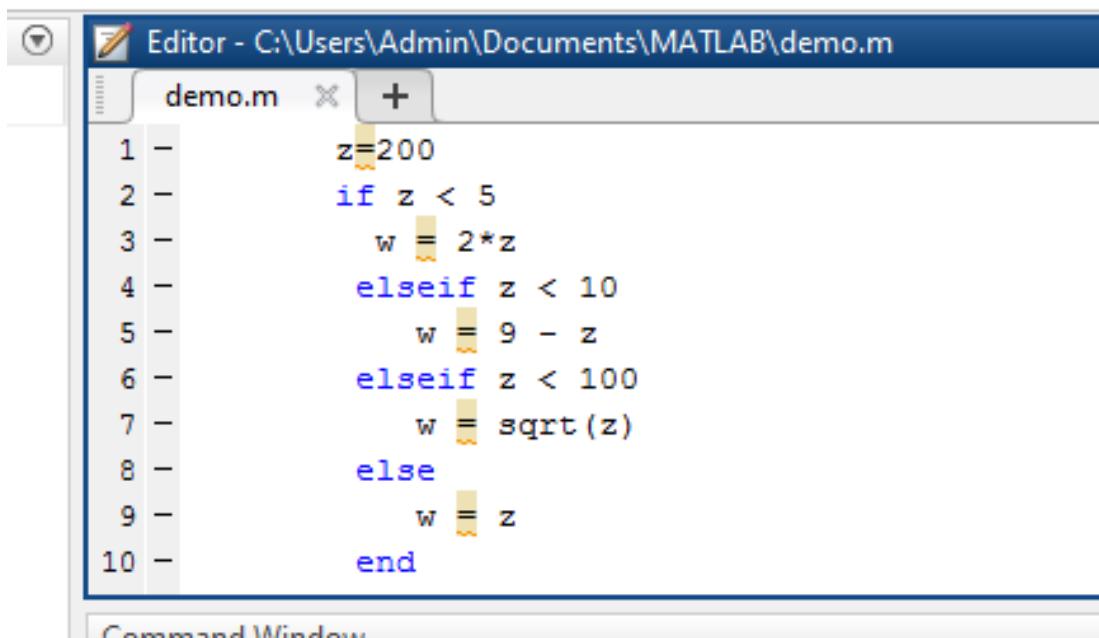
z =
60

w =
7.7460

fx >>
```

d. $z = 200$ $w = ?$

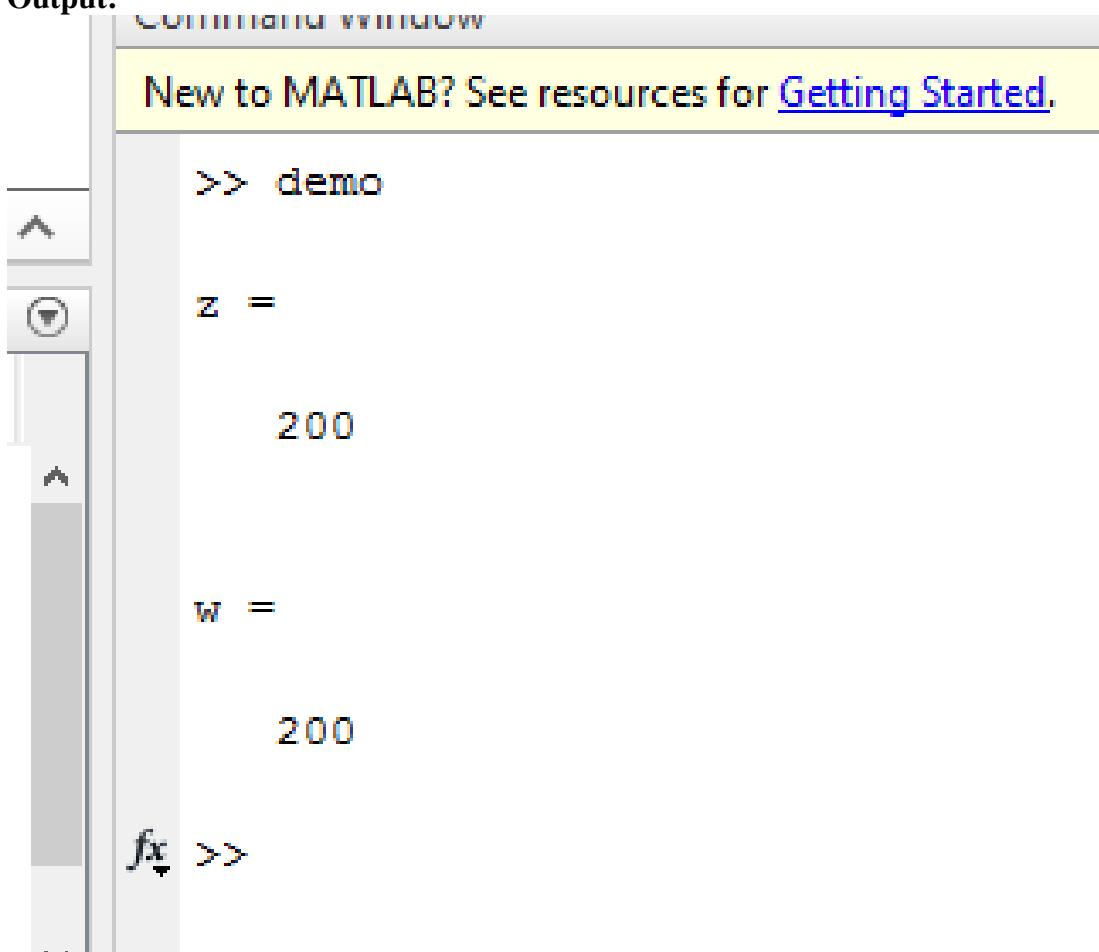
Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code defines a variable z and calculates w based on its value. The code is as follows:

```
1 - z=200
2 - if z < 5
3 -     w = 2*z
4 - elseif z < 10
5 -     w = 9 - z
6 - elseif z < 100
7 -     w = sqrt(z)
8 - else
9 -     w = z
10 - end
```

Output:



The screenshot shows the MATLAB Command Window. After running the script with the command `>> demo`, the output shows the value of z and w .

```
>> demo
z =
200
w =
200
fx >>
```

```

3. if T < 30
   h = 2*T + 1
elseif T < 10
   h = T - 2
else
   h = 0
end

```

a.. T = 50 h = ?

Code:

```

Editor - C:\Users\Admin\Documents\MATLAB\demo.m
demo.m  X  +
1 -      T=50
2 -      if T < 30
3 -          h = 2*T + 1
4 -      elseif T < 10
5 -          h = T - 2
6 -      else
7 -          h = 0
8 -      end
9

```

Output:

```

New to MATLAB? See resources for Getting Started.
>> demo

T =
50

h =
0

fx >>

```

b. $T = 15$ $h = ?$

Code:

The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code defines a variable T and uses an if-else-if-else structure to calculate h based on the value of T.

```
1 - T=15
2 - if T < 30
3 -     h = 2*T + 1
4 - elseif T < 10
5 -     h = T - 2
6 - else
7 -     h = 0
8 - end
```

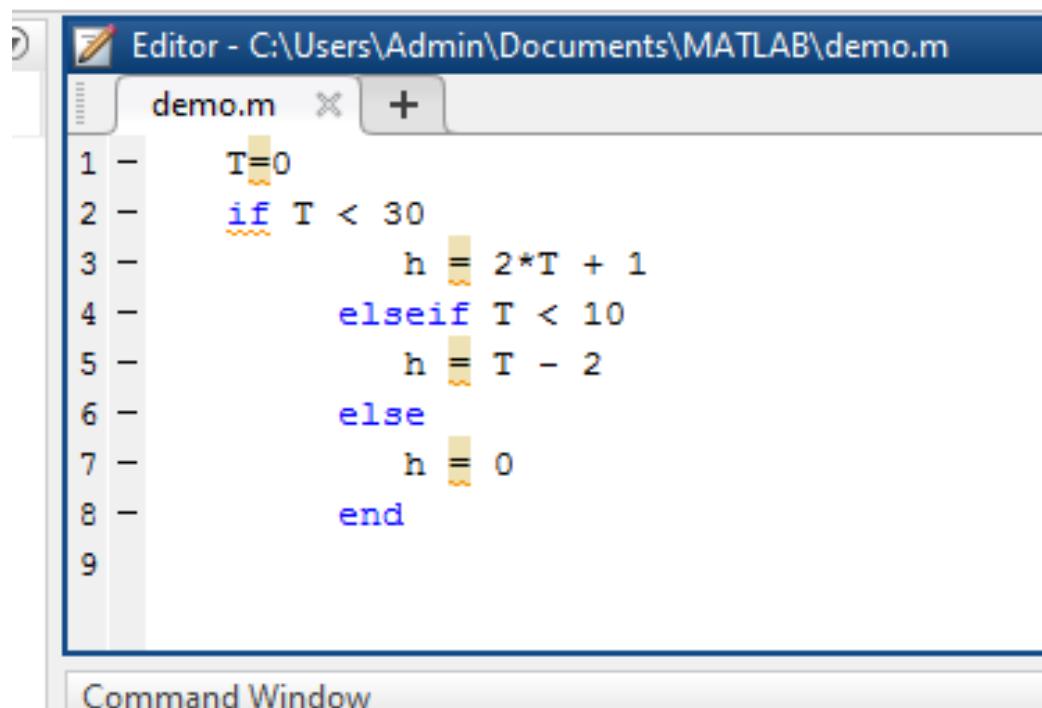
Output:

The screenshot shows the MATLAB Command Window. The user types 'demo' and presses Enter. The program outputs the value of T (15) and the calculated value of h (31).

```
>> demo
T =
15
h =
31
fx >>
```

c. $T = 0 \quad h = ?$

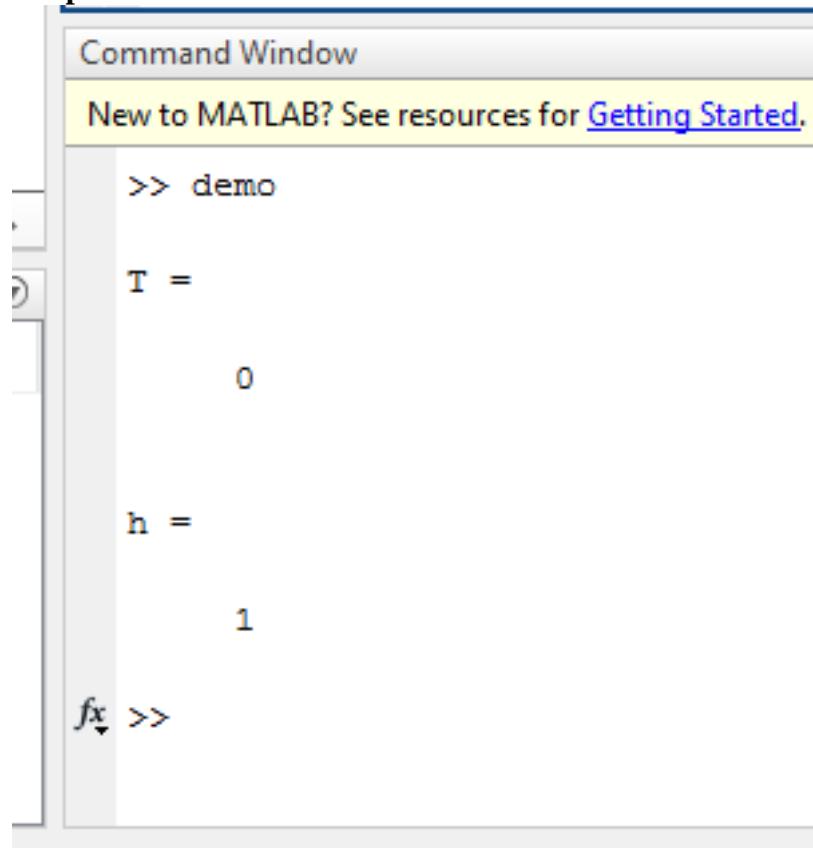
Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code defines a variable $T=0$ and uses an if-elseif-else structure to calculate h based on the value of T . The code is as follows:

```
1 - T=0
2 - if T < 30
3 -     h = 2*T + 1
4 - elseif T < 10
5 -     h = T - 2
6 - else
7 -     h = 0
8 - end
9 -
```

Output:



The screenshot shows the MATLAB Command Window. After running the script 'demo', the variables T and h are displayed with their values.

```
Command Window
New to MATLAB? See resources for Getting Started.
>> demo
T =
0
h =
1
fx >>
```

```

4. if 0 < x < 10      a. x = -1  y = ?
    y = 4*x            b. x = 5   y = ?
    elseif 10 < x < 40  c. x = 30  y = ?
    y = 10*x            d. x = 100 y = ?
    else
        y = 500
    end

```

a. x = -1 y = ?

Code:

The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code in the editor is:

```

1 - x=-1
2 - if 0 < x < 10
3 -     y = 4*x
4 - elseif 10 < x < 40
5 -     y = 10*x
6 - else
7 -     y = 500
8 -
9

```

Output:

The screenshot shows the MATLAB Command Window. After running the command `>> demo`, the output is:

```

x =
-1

y =
-4

```

b. $x = 5$ $y = ?$

Code:

The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code defines a variable $x=5$ and uses an if-elseif-else structure to calculate y based on the value of x . The code is as follows:

```
demo.m
x=5
if 0 < x < 10
    y = 4*x
elseif 10 < x < 40
    y = 10*x
else
    y = 500
end
```

Output:

The screenshot shows the MATLAB Command Window. After running the command `>> demo`, the output shows the value of x is 5 and the calculated value of y is 20.

```
Command Window
>> demo
x =
5
y =
20
fx >>
```

c. . x = 30 y = ?

Code:

The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code defines a variable 'x' and uses an if-elseif-else structure to calculate 'y' based on the value of 'x'. The code is as follows:

```
1 - x=30
2 - if 0 < x < 10
3 -     y = 4*x
4 - elseif 10 < x < 40
5 -     y = 10*x
6 - else
7 -     y = 500
8 - end
```

Output:

The screenshot shows the MATLAB Command Window. It starts with a welcome message for new users, followed by the execution of the 'demo' script. The user inputs 'x = 30' and the script outputs 'y = 120'. The command prompt 'fx >>' is shown at the bottom.

```
Command Window
New to MATLAB? See resources for Getting Started.
>> demo

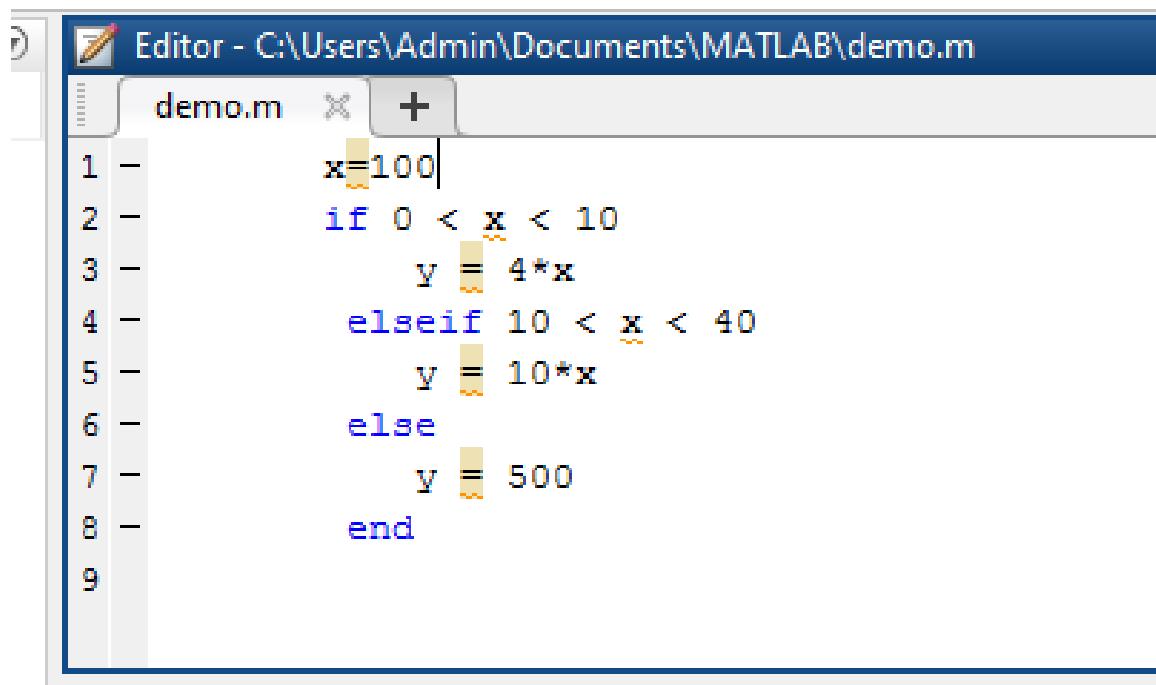
x =
30

y =
120

fx >>
```

d. . x = 100 y = ?

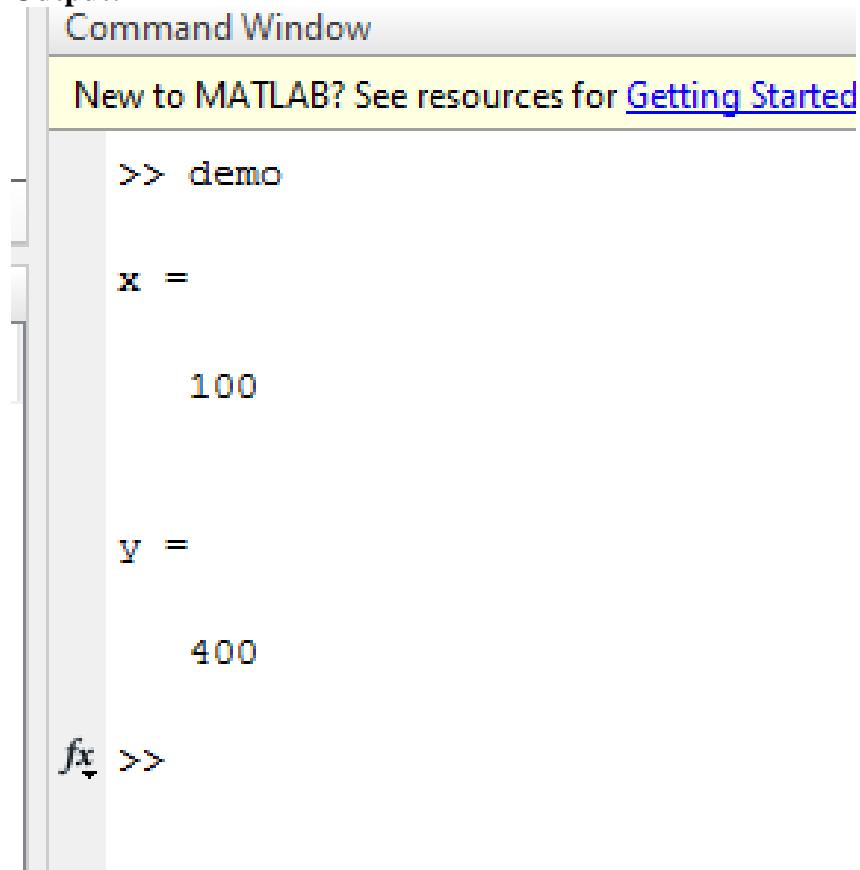
Code:



The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code defines a variable `x` and calculates `y` based on its value using an if-elseif-else structure.

```
1 - x=100
2 - if 0 < x < 10
3 -     y = 4*x
4 - elseif 10 < x < 40
5 -     y = 10*x
6 - else
7 -     y = 500
8 - end
9 -
```

Output:



The screenshot shows the MATLAB Command Window. After running the script with `>> demo`, it displays the value of `x` and the calculated value of `y`.

```
Command Window
New to MATLAB? See resources for Getting Started
>> demo
x =
100
y =
400
fx >>
```

5. Given that $x = [1 \ 5 \ 2 \ 8 \ 9 \ 0 \ 1 \ 6 \ 7]$ and $y = [5 \ 2 \ 2 \ 6 \ 0 \ 0 \ 2 \ 7 \ 3]$, execute and explain the results of the following commands:

a. $x > y$

Code:

The screenshot shows the MATLAB Editor window with a file named 'demo.m'. The code contains three lines: 'x = [1 5 2 8 9 0 1 6 7]', 'y = [5 2 2 6 0 0 2 7 3]', and 'x > y'. The variable 'x' is highlighted in yellow, and the comparison operator '>' is highlighted in orange.

Output:

The screenshot shows the MATLAB Command Window. It displays the output of running the 'demo' script. The variables 'x' and 'y' are defined as row vectors. The variable 'ans' is a logical array of size 1x9, representing the result of the element-wise comparison 'x > y'. The values in 'ans' are 0, 1, 0, 1, 1, 0, 0, 0, 1, indicating that elements at indices 1, 4, 5, 6, and 9 of 'x' are greater than the corresponding elements in 'y'.

```
>> demo

x =
    1     5     2     8     9     0     1     6     7

y =
    5     2     2     6     0     0     2     7     3

ans =
 1×9 logical array
 0     1     0     1     1     0     0     0     1
```

b. $y > x$

Code:

The screenshot shows the MATLAB Editor window with the file 'demo.m' open. The code contains three lines: 'x = [1 5 2 8 9 0 1 6 7]', 'y = [5 2 2 6 0 0 2 7 3]', and 'y > x'. Below the editor is the Command Window.

```
Editor - C:\Users\Admin\Documents\MATLAB\demo.m
demo.m + 
1 - x = [1 5 2 8 9 0 1 6 7]
2 - y = [5 2 2 6 0 0 2 7 3]
3 - y > x

Command Window
```

Output:

The screenshot shows the MATLAB Command Window. It displays the output of running the 'demo' script, which includes the definition of arrays 'x' and 'y', and the result of the comparison 'y > x'.

```
Command Window
New to MATLAB? See resources for Getting Started.
>> demo

x =
1 5 2 8 9 0 1 6 7

y =
5 2 2 6 0 0 2 7 3

ans =
1×9 logical array
1 0 0 0 0 0 1 1 0

fx >>
```

c. $x == y$

Code:

The screenshot shows the MATLAB Editor window with a file named 'demo.m'. The code contains three lines of MATLAB code:

```
1 - x = [1 5 2 8 9 0 1 6 7]
2 - y = [5 2 2 6 0 0 2 7 3]
3 - y == x
```

Output:

The screenshot shows the MATLAB Command Window with the following output:

```
New to MATLAB? See resources for Getting Started.
>> demo

x =
    1     5     2     8     9     0     1     6     7

y =
    5     2     2     6     0     0     2     7     3

ans =
1×9 logical array
    0     0     1     0     0     1     0     0     0

fx >>
```

d. $x \leq y$

Code:

► MATLAB ►

The screenshot shows the MATLAB Editor window with two files open: 'demo.m' and 'demoo.m'. The 'demoo.m' file is active and contains the following code:

```
1 - x = [1 5 2 8 9 0 1 6 7]
2 - y = [5 2 2 6 0 0 2 7 3]
3 - x <= y
4
```

Output:

The screenshot shows the MATLAB Command Window output for the code in 'demoo.m'. The output is as follows:

```
x =
1 5 2 8 9 0 1 6 7

y =
5 2 2 6 0 0 2 7 3

ans =
1×9 logical array
1 0 1 0 0 1 1 1 0
```

e. $y \leq x$

Code:

The screenshot shows the MATLAB Editor window with two files open: 'demo.m' and 'demoo.m'. The 'demoo.m' file contains the following code:

```
1 -      x = [1 5 2 8 9 0 1 6 7]
2 -      y = [5 2 2 6 0 0 2 7 3]
3 -      y <= x
4
```

Output:

The screenshot shows the MATLAB Command Window output for the code in 'demoo.m'. The output is:

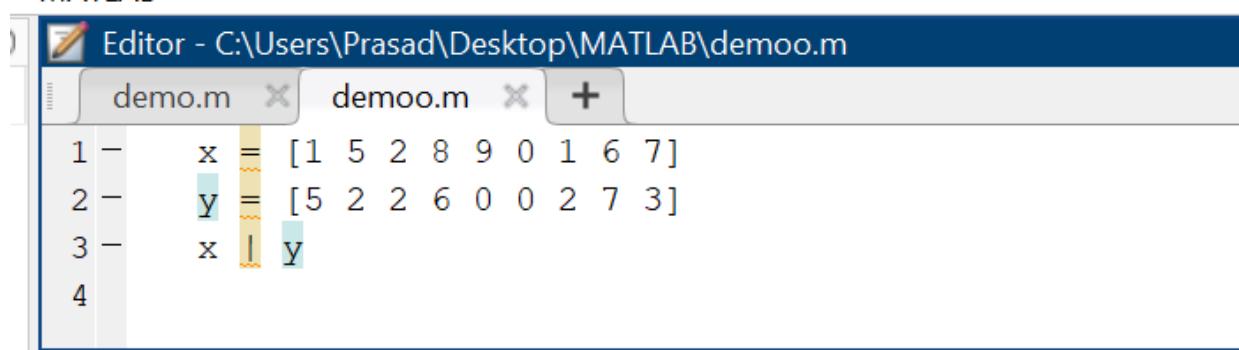
```
x =
1 5 2 8 9 0 1 6 7

y =
5 2 2 6 0 0 2 7 3

ans =
1×9 logical array
0 1 1 1 1 1 0 0 1
```

f. $x | y$

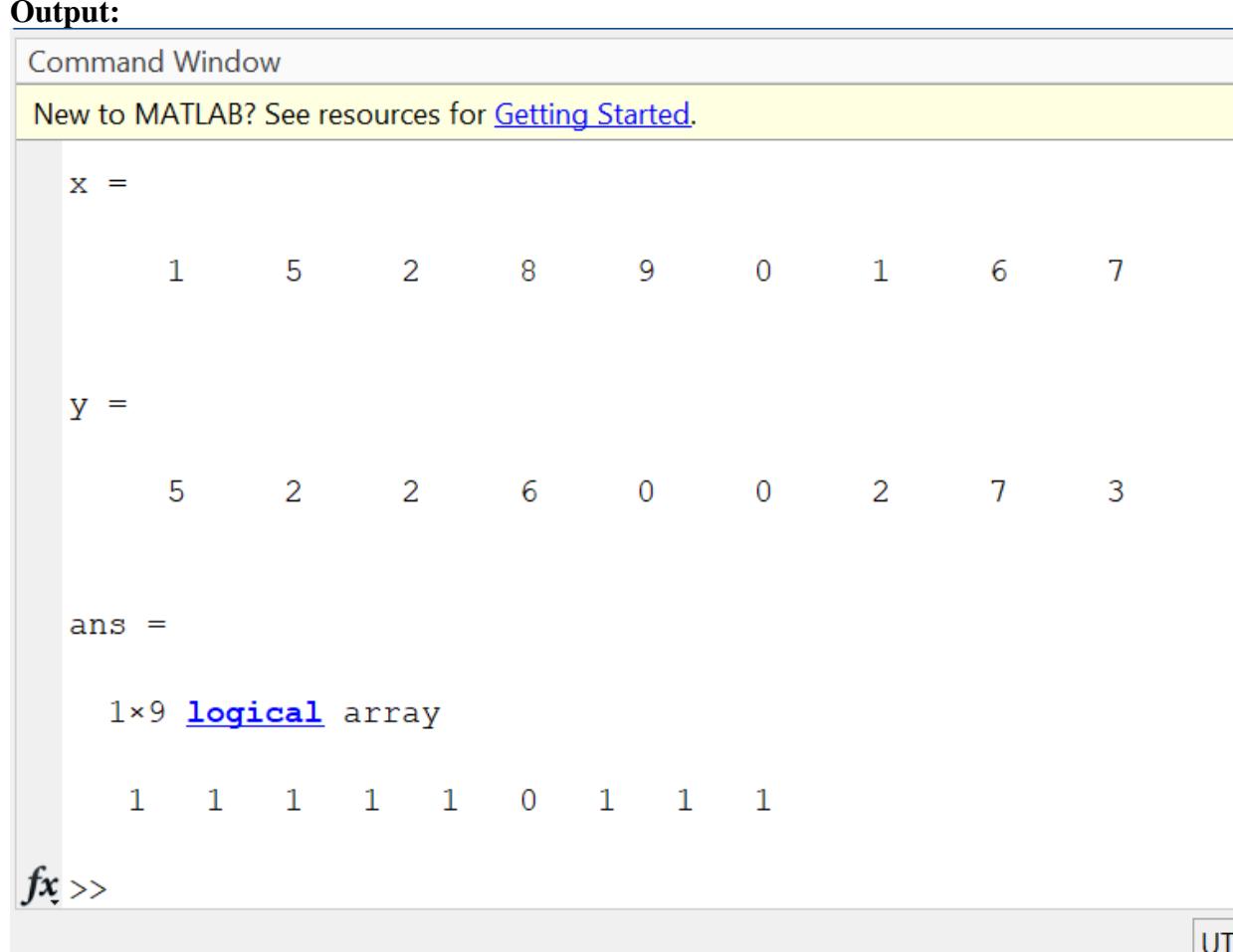
Code:



The screenshot shows the MATLAB Editor window. The title bar says "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". There are two tabs open: "demo.m" and "demoo.m". The "demoo.m" tab is active and contains the following code:

```
1 - x = [1 5 2 8 9 0 1 6 7]
2 - y = [5 2 2 6 0 0 2 7 3]
3 - x | y
4
```

Output:



The screenshot shows the MATLAB Command Window. It displays the output of the code from the previous screenshot. The output is:

```
x =
1 5 2 8 9 0 1 6 7

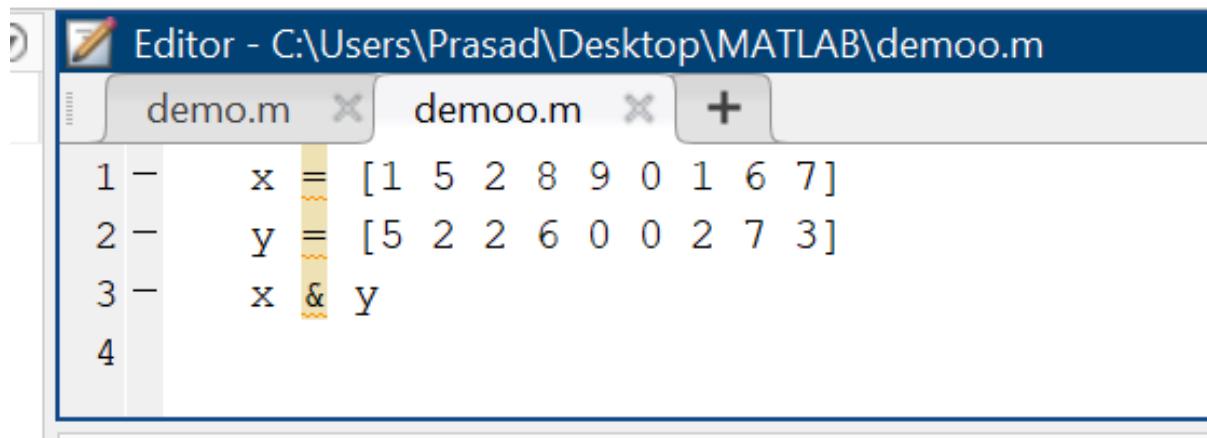
y =
5 2 2 6 0 0 2 7 3

ans =
1×9 logical array
1 1 1 1 1 0 1 1 1
```

g. x & y

Code:

► MATLAB ►



Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m demoo.m +

1 - x = [1 5 2 8 9 0 1 6 7]
2 - y = [5 2 2 6 0 0 2 7 3]
3 - x & y
4

Output:



Command Window

New to MATLAB? See resources for [Getting Started](#).

```
x =
1      5      2      8      9      0      1      6      7

y =
5      2      2      6      0      0      2      7      3

ans =
1×9 logical array

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

fx >>

h. $x \& (\sim y)$

Code:

The screenshot shows the MATLAB Editor window with two files open: 'demo.m' and 'demoo.m'. The 'demoo.m' file contains the following code:

```
1 - x = [1 5 2 8 9 0 1 6 7]
2 - y = [5 2 2 6 0 0 2 7 3]
3 - x & (~y)
```

Command Window

Output:

```
Command Window
New to MATLAB? See resources for Getting Started.
x =
1 5 2 8 9 0 1 6 7

y =
5 2 2 6 0 0 2 7 3

ans =
1×9 logical array
0 0 0 0 1 0 0 0 0
fx >>
```

i. $(x > y) \mid (y > x)$

Code:

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 -      x = [1 5 2 8 9 0 1 6 7]
2 -      y = [5 2 2 6 0 0 2 7 3]
3 -      |(x > y) | (y > x)
```

Output:

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
x =
1 5 2 8 9 0 1 6 7

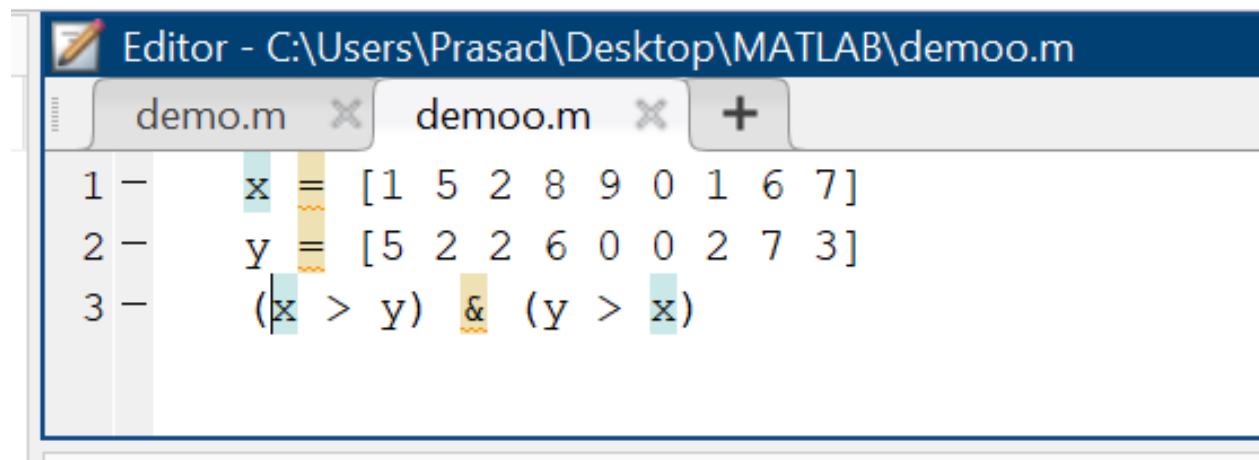
y =
5 2 2 6 0 0 2 7 3

ans =
1×9 logical array
1 1 0 1 1 0 1 1 1
fr <<
```

j. $(x > y) \& (y > x)$

Code:

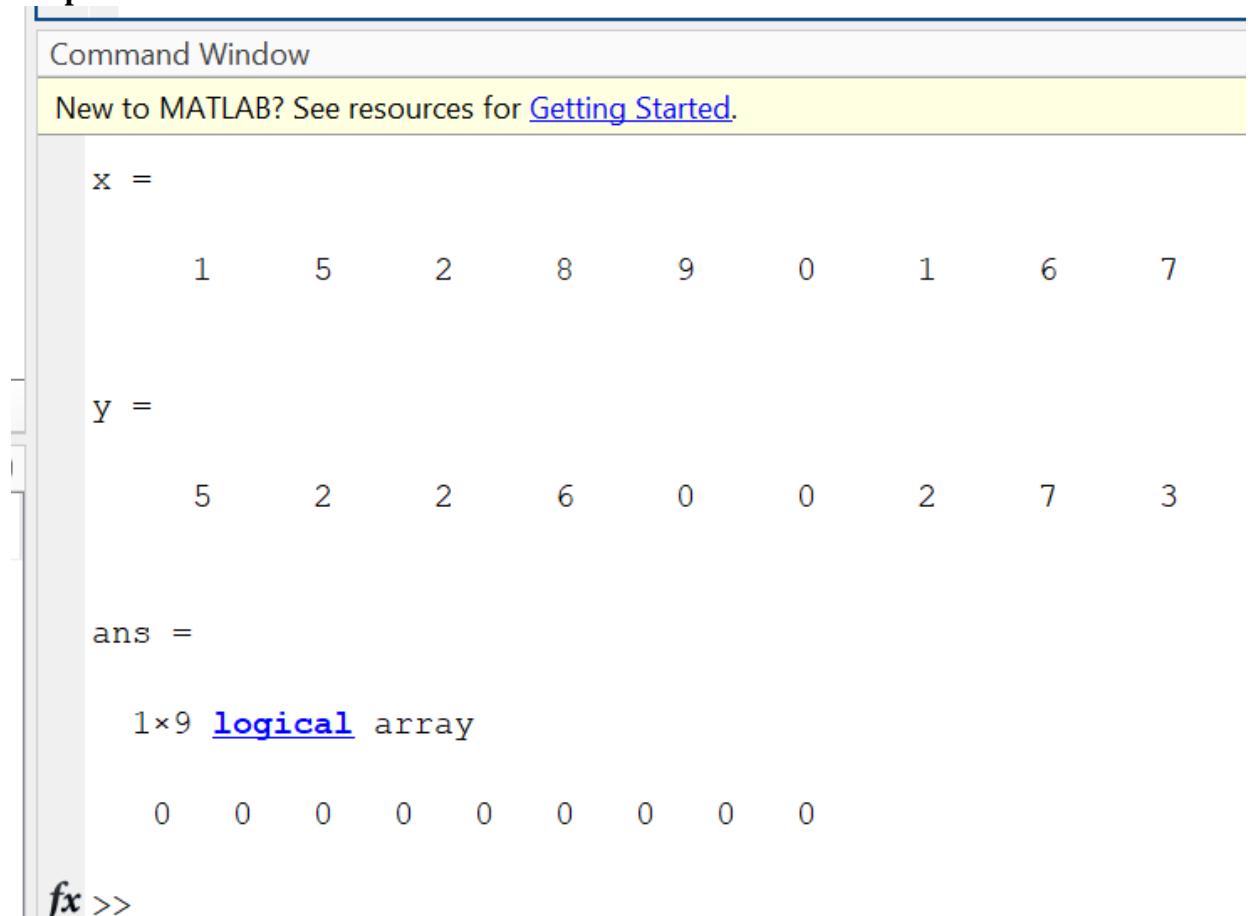
· MATLAB ▶



The screenshot shows the MATLAB Editor window with two files open: 'demo.m' and 'demoo.m'. The 'demo.m' file contains the following code:

```
1 - x = [1 5 2 8 9 0 1 6 7]
2 - y = [5 2 2 6 0 0 2 7 3]
3 - (x > y) & (y > x)
```

Output:



The screenshot shows the MATLAB Command Window with the following output:

```
x =
1 5 2 8 9 0 1 6 7

y =
5 2 2 6 0 0 2 7 3

ans =
1×9 logical array
0 0 0 0 0 0 0 0 0
```

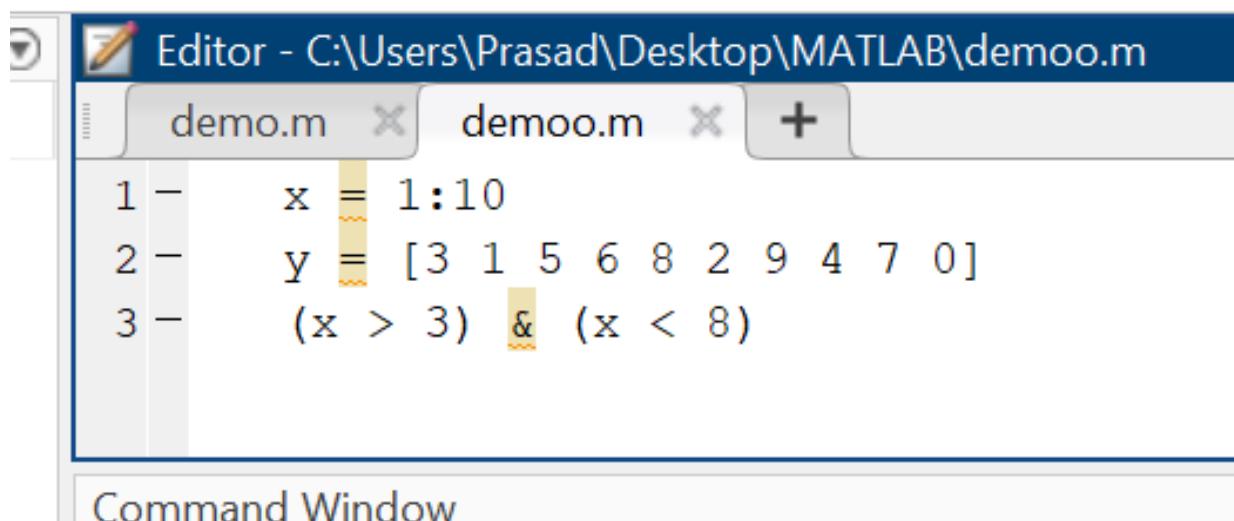
fx >>

6. The exercises here show the techniques of logical-indexing (indexing with 0-1 vectors). Given $x = 1:10$ and $y = [3 \ 1 \ 5 \ 6 \ 8 \ 2 \ 9 \ 4 \ 7 \ 0]$, execute and interpret the results of the following commands:

a. $(x > 3) \ \& \ (x < 8)$

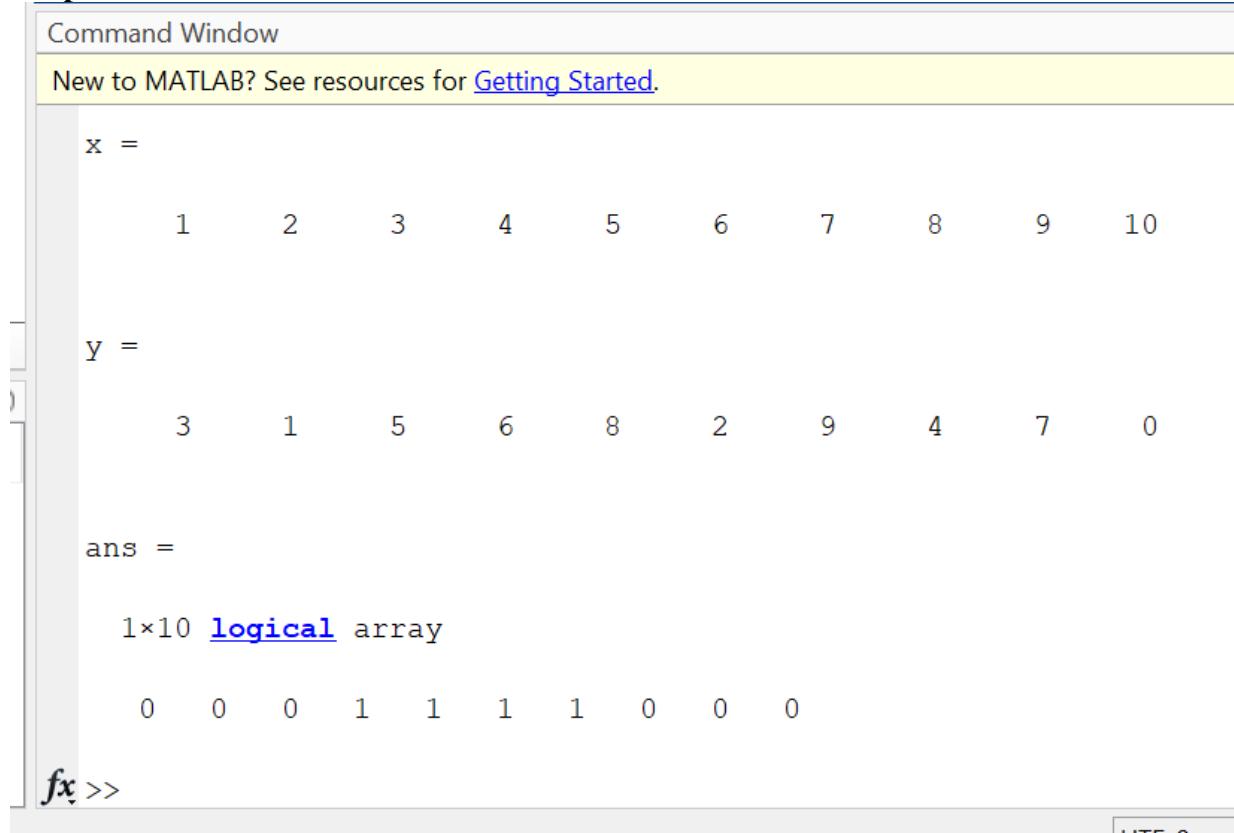
Code:

► MATLAB ►



Command Window

Output:



b.x(x > 5)

Code:

· MATLAB ▶

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m demoo.m +

```
1 - x = 1:10
2 - y = [3 1 5 6 8 2 9 4 7 0]
3 - x(x > 5)
```

Output:

Command Window

New to MATLAB? See resources for [Getting Started.](#)

```
>> demoo
```

```
x =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
y =
```

```
3 1 5 6 8 2 9 4 7 0
```

```
ans =
```

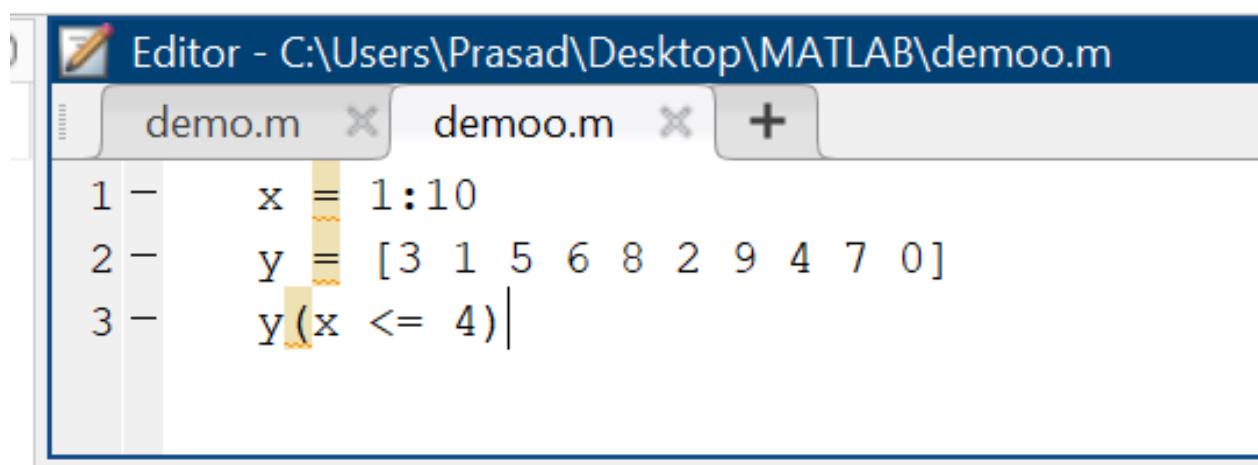
```
6 7 8 9 10
```

fx >>

c. $y(x \leq 4)$

Code:

MATLAB



The screenshot shows the MATLAB Editor window with two files open: 'demo.m' and 'demoo.m'. The current file is 'demoo.m'. The code in 'demoo.m' is as follows:

```
1 -      x = 1:10
2 -      y = [3 1 5 6 8 2 9 4 7 0]
3 -      y(x <= 4)
```

Output:

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> demoo

x =
1 2 3 4 5 6 7 8 9 10

y =
3 1 5 6 8 2 9 4 7 0

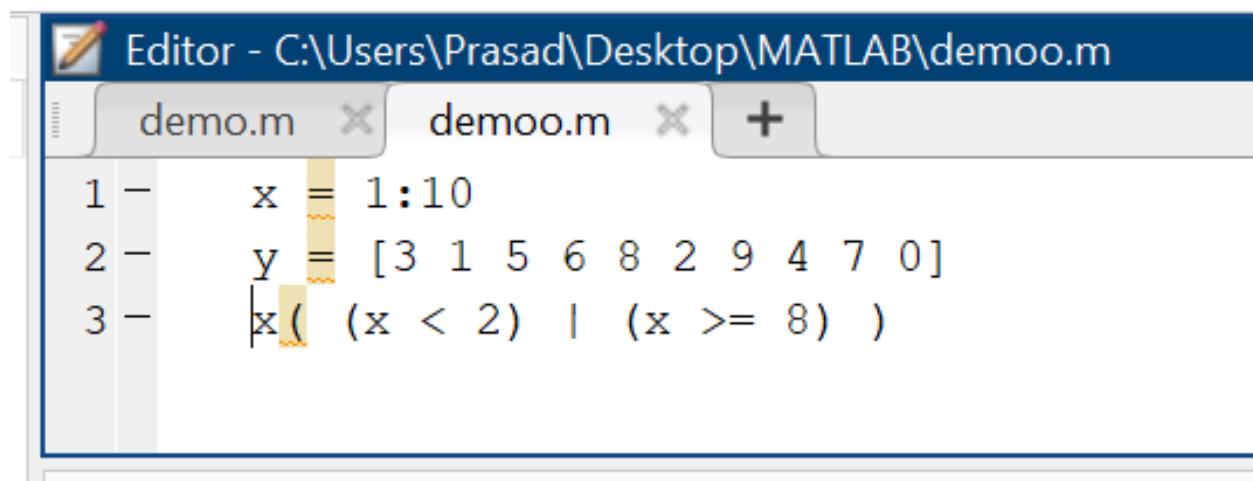
ans =
3 1 5 6

fx >>
```

`d.x((x < 2) | (x >= 8))`

Code:

MATLAB ▶



The screenshot shows the MATLAB Editor window with two tabs: "demo.m" and "demoo.m". The "demoo.m" tab is active, displaying the following code:

```
1 - x = 1:10
2 - y = [3 1 5 6 8 2 9 4 7 0]
3 - x( (x < 2) | (x >= 8) )
```

Output:

New to MATLAB? See resources for [Getting Started](#).

```
>> demoo

x =
1 2 3 4 5 6 7 8 9 10

y =
3 1 5 6 8 2 9 4 7 0

ans =
1 8 9 10

fx >>
```

e. $y((x < 2) | (x \geq 8))$

Code:

MATLAB ▶

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 -      x = 1:10
2 -      y = [3 1 5 6 8 2 9 4 7 0]
3 -      y( (x < 2) | (x >= 8) )
```

Output:

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> demoo
```

```
x =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
y =
```

```
3 1 5 6 8 2 9 4 7 0
```

```
ans =
```

```
3 4 7 0
```

```
fx >>
```

f. $x(y < 0)$

Code:



The screenshot shows the MATLAB Editor window titled "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". It contains two tabs: "demo.m" and "demoo.m". The "demoo.m" tab is active and displays the following code:

```
1 - x = 1:10
2 - y = [3 1 5 6 8 2 9 4 7 0]
3 - x(y < 0)
```

Output:

New to MATLAB? See resources for [Getting Started](#).

```
>> demoo

x =
1 2 3 4 5 6 7 8 9 10

y =
3 1 5 6 8 2 9 4 7 0

ans =
1×0 empty double row vector

fx >>
```

Assignment 4

Date of Submission: 12/08/2022

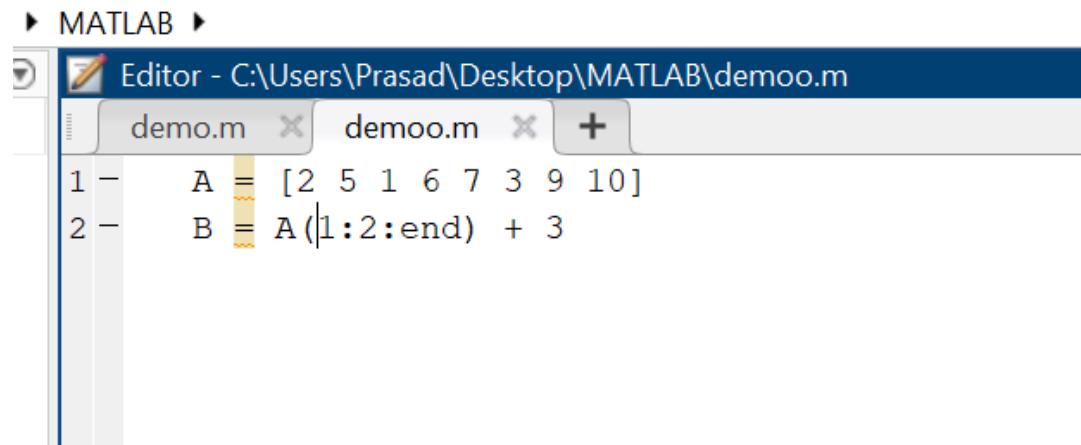
In-built Functions of MATLAB

1. $A = [2 \ 5 \ 1 \ 6 \ 7 \ 3 \ 9 \ 10]$, add 3 to the odd indexed numbers.

Explanation:

By using parenthesis and colon operator access the index required that and add 3,
 $B=A(1:2:end)+3$

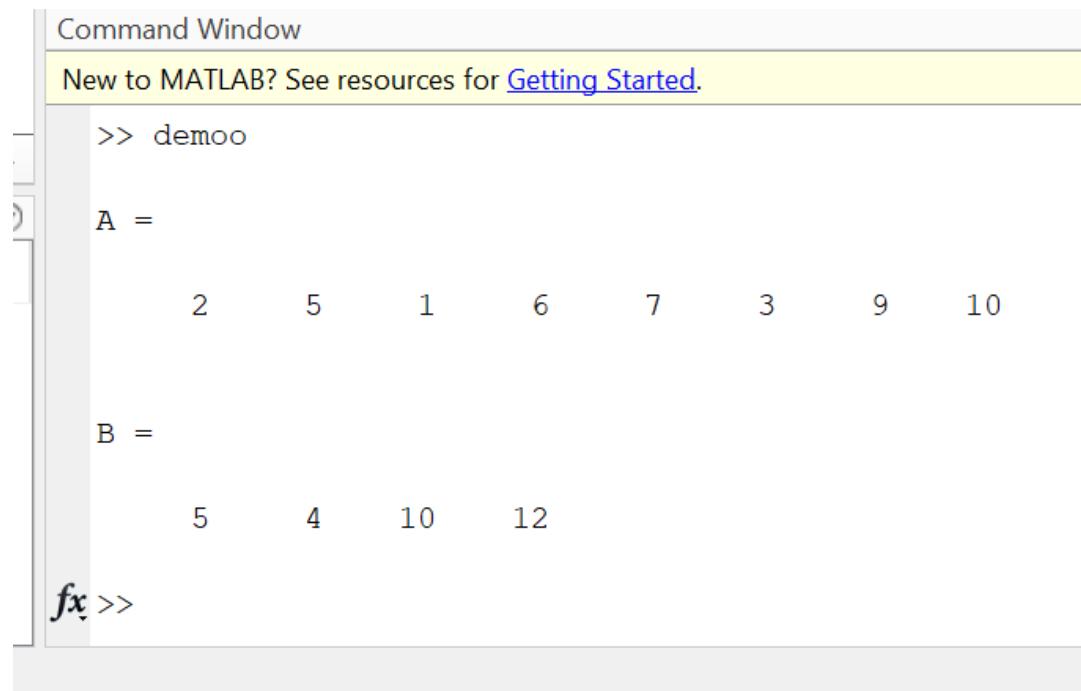
Code:



The screenshot shows the MATLAB Editor window. The title bar says "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". There are two tabs: "demo.m" and "demoo.m". The "demoo.m" tab is active, displaying the following code:

```
1 - A = [2 5 1 6 7 3 9 10]
2 - B = A(1:2:end) + 3
```

Output:



The screenshot shows the MATLAB Command Window. It displays the output of running the script "demoo.m". The output is:

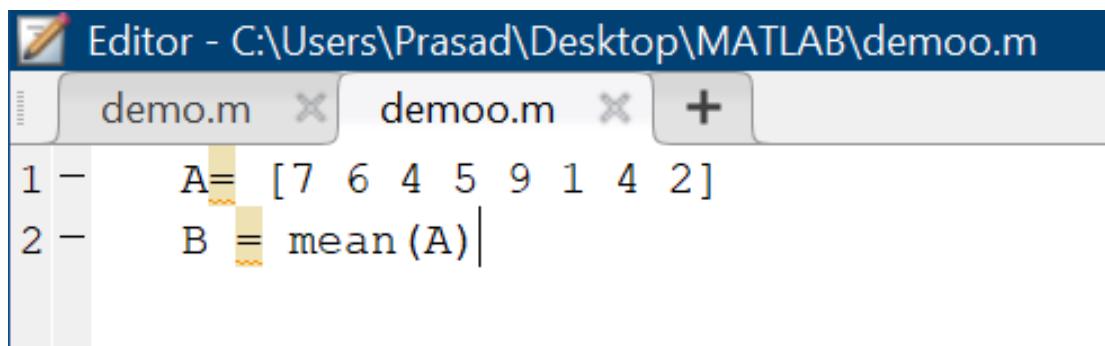
```
New to MATLAB? See resources for Getting Started.
>> demoo
A =
    2     5     1     6     7     3     9     10
B =
    5     4    10    12
fx >>
```

2. Create a row vector [7 6 4 5 9 1 4 2] and calculate the mean of it.

Explanation:

Use the function mean() to calculate the mean of the row vector.

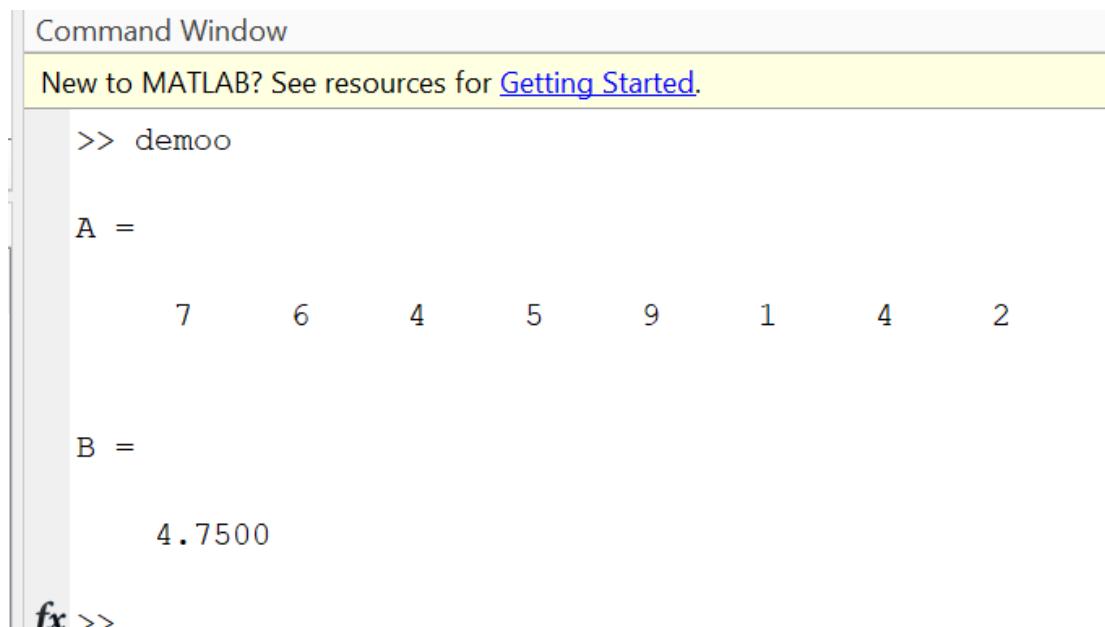
Code:



The screenshot shows the MATLAB Editor window. It has two tabs open: 'demo.m' and 'demoo.m'. The 'demoo.m' tab is active and contains the following code:

```
1 - A = [7 6 4 5 9 1 4 2]
2 - B = mean(A)
```

Output:



The screenshot shows the MATLAB Command Window. It displays the output of running the 'demoo' script. The output is:

```
>> demoo

A =
    7     6     4     5     9     1     4     2

B =
4.7500

fx >>
```

3. Evaluate the following MATLAB expressions by hand and use MATLAB to check the answers. Explain **round()**, **floor()** and **ceil()**.

i. $2 / 2 * 3$

Explanation:

`round()` rounds each element of X to the nearest integer

`ceil()` rounds to the next higher integer, in the +infinity direction.

`floor()` rounds to the next lower integer, in the -infinity direction.

Code:

MATLAB ▶

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m demoo.m +
1 - A = 2 / 2 * 3
2 - B = round(A)
```

MATLAB ▶

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m demoo.m +
1 - A = 2 / 2 * 3
2 - B = floor(A)
```

MATLAB ▶

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

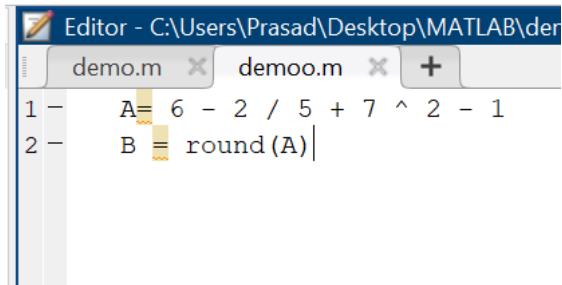
```
demo.m demoo.m +
1 - A = 2 / 2 * 3
2 - B = ceil(A)
```

Output:

```
Command Window
New to MATLAB? See resources for Getting Started.
>> demoo
A =
3
B =
3
Command Window
New to MATLAB? See resources for Getting Star
>> demoo
A =
3
B =
3
fx >>
Command Window
New to MATLAB? See resources for Getting
>> demoo
A =
3
B =
3
fx >>
```

ii. $6 - 2 / 5 + 7^2 - 1$

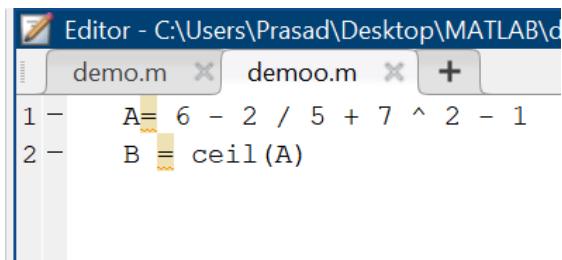
Code:



Editor - C:\Users\Prasad\Desktop\MATLAB\der

```
demo.m      demoo.m      +
```

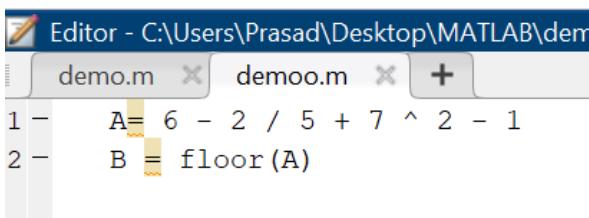
```
1 - A = 6 - 2 / 5 + 7 ^ 2 - 1
2 - B = round(A)
```



Editor - C:\Users\Prasad\Desktop\MATLAB\c

```
demo.m      demoo.m      +
```

```
1 - A = 6 - 2 / 5 + 7 ^ 2 - 1
2 - B = ceil(A)
```

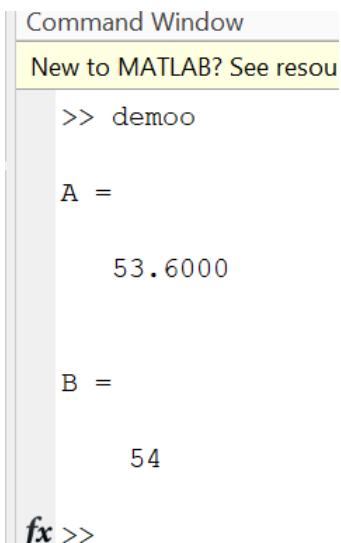


Editor - C:\Users\Prasad\Desktop\MATLAB\dem

```
demo.m      demoo.m      +
```

```
1 - A = 6 - 2 / 5 + 7 ^ 2 - 1
2 - B = floor(A)
```

Output:



Command Window

New to MATLAB? See resources & help.

```
>> demoo
```

```
A =
```

```
53.6000
```

```
B =
```

```
54
```

```
fx >>
```

Command Window
New to MATLAB? See res

```
>> demo0
```

```
A =
```

```
53.6000
```

```
B =
```

```
53
```

fx >>

Command Window
New to MATLAB? See res

```
>> demo0
```

```
A =
```

```
53.6000
```

```
B =
```

```
54
```

fx >>

$$\text{iii. } 10 / 2 \backslash 5 - 3 + 2 * 4$$

Code:

The image shows three separate MATLAB code editors side-by-side, each containing a different line of code to calculate the expression $10 / 2 \backslash 5 - 3 + 2 * 4$.

- The first editor shows: `A = 10 / 2 \ 5 - 3 + 2 * 4`
- The second editor shows: `B = ceil(A)`
- The third editor shows: `B = round(A)`

Output:

The image shows the MATLAB Command Window displaying the output of the code from the previous section.

```
Command Window
>> demoo

A =
6

B =
6

fx >>
```

```
>> demoo
```

```
A =
```

```
6
```

```
B =
```

```
6
```

```
x>>
```

```
>> demoo
```

```
A =
```

```
6
```

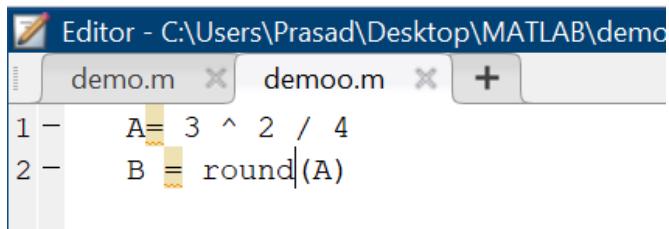
```
B =
```

```
6
```

```
x>>
```

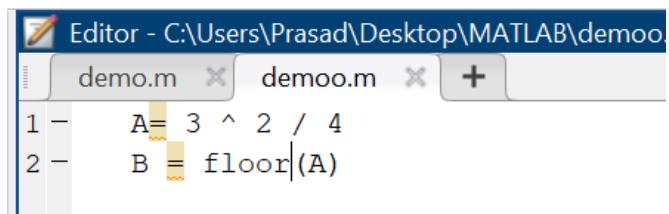
iv. $3^{2/4}$

Code:



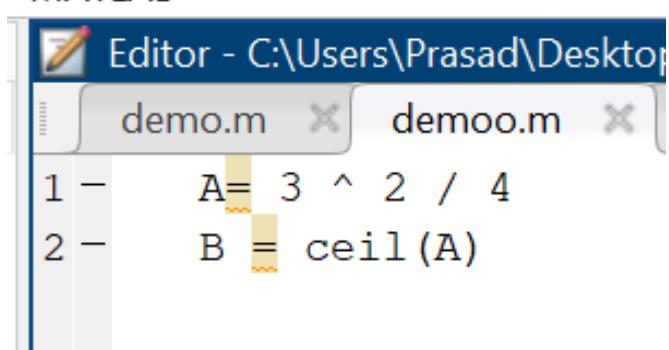
```
Editor - C:\Users\Prasad\Desktop\MATLAB\demo
demo.m demoo.m +
```

```
1 - A = 3 ^ 2 / 4
2 - B = round(A)
```



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo
demo.m demoo.m +
```

```
1 - A = 3 ^ 2 / 4
2 - B = floor(A)
```



```
Editor - C:\Users\Prasad\Desktop
demo.m demoo.m +
```

```
1 - A = 3 ^ 2 / 4
2 - B = ceil(A)
```

Output:

```
A =
2.2500

B =
2

fx >>
```

```
>> demo0

A =
2.2500

B =
2

fx >>

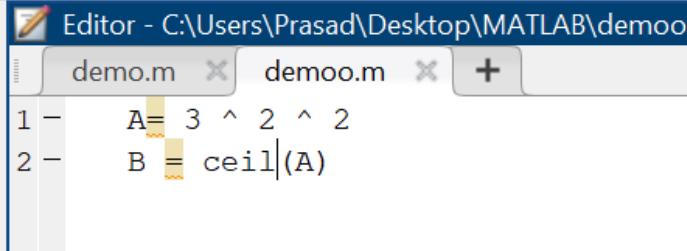
A =
2.2500

B =
3

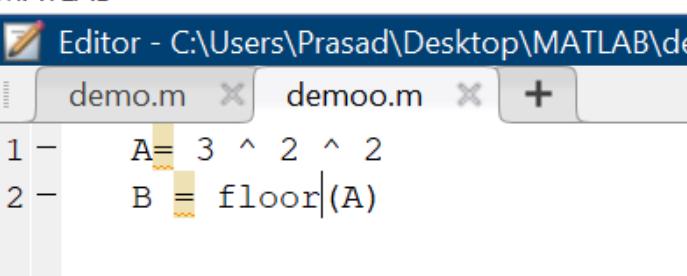
fx >>
```

v. 3^2^2

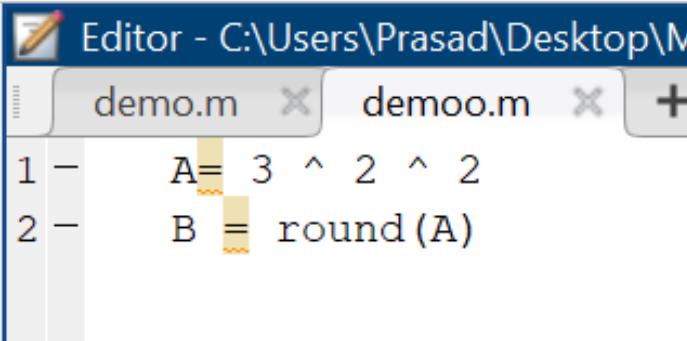
Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m demoo.m +  
1 - A = 3 ^ 2 ^ 2  
2 - B = ceil(A)
```



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m demoo.m +  
1 - A = 3 ^ 2 ^ 2  
2 - B = floor(A)
```



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m demoo.m +  
1 - A = 3 ^ 2 ^ 2  
2 - B = round(A)
```

Output:

```
>> demoo

A =

    81

B =

    81

fx >>
```

```
>> demoo
```

```
A =
```

```
81
```

```
B =
```

```
81
```

fx >>

```
>> demoo
```

```
A =
```

```
81
```

```
B =
```

```
81
```

fx >>

vi. $2 + \text{round}(6 / 9 + 3 * 2) / 2 - 3$

Code:

The figure consists of three vertically stacked screenshots of the MATLAB Editor. Each screenshot shows a file named 'demo.m' with the following code:

```
1 - A = 2 + round(6 / 9 + 3 * 2) / 2 - 3
2 - B = round(A)
```

In the first screenshot, the cursor is at the end of the second line. In the second, it's in the middle of the second line. In the third, it's at the end of the first line.

Output:

```
>> demoo

A =
2.5000

B =
3

fz --
```

```
>> demoo

A =
2.5000

B =
2

fx >>
```

```
>> demo0  
  
A =  
  
    2.5000  
  
B =  
  
    3  
  
fx >>
```

$$\text{vii. } 2 + \text{floor}(6 / 9 + 3 * 2) / 2 - 3$$

Code:

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m demo.m +
1 - A = 2 + floor(6 / 9 + 3 * 2) / 2 - 3
2 - B = floor(A)
```

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m demo.m +
1 - A = 2 + floor(6 / 9 + 3 * 2) / 2 - 3
2 - B = ceil(A)
```

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m demo.m +
1 - A = 2 + floor(6 / 9 + 3 * 2) / 2 - 3
2 - B = round(A)
```

Output:

```
>> demoo

A =
2

B =
2

fx >>
```

```
>> demoo

A =
2

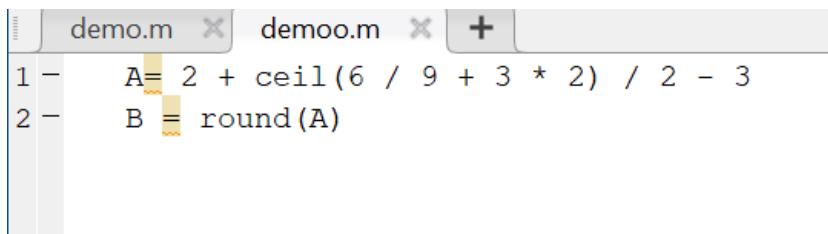
B =
2

fx >>
```

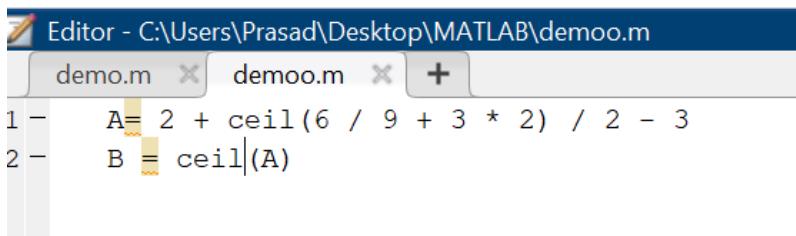
```
>> demoo  
  
A =  
  
2  
  
B =  
  
2  
  
fx >>
```

$$\text{viii. } 2 + \lceil 6 / 9 + 3 * 2 \rceil / 2 - 3$$

Code:

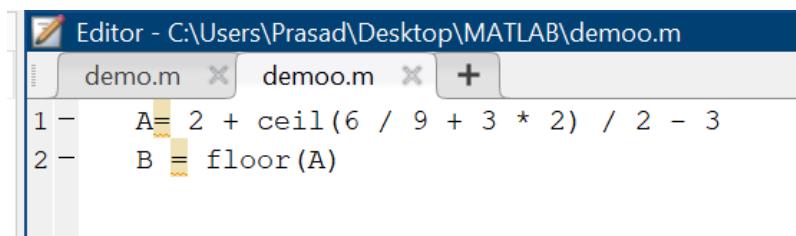


```
demo.m x demoo.m x +  
1 - A = 2 + ceil(6 / 9 + 3 * 2) / 2 - 3  
2 - B = round(A)
```



Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

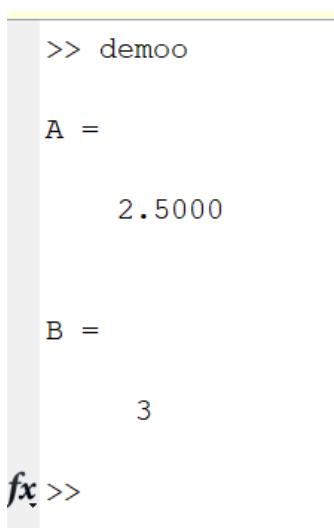
```
demo.m x demoo.m x +  
1 - A = 2 + ceil(6 / 9 + 3 * 2) / 2 - 3  
2 - B = ceil(A)
```



Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m x demoo.m x +  
1 - A = 2 + ceil(6 / 9 + 3 * 2) / 2 - 3  
2 - B = floor(A)
```

Output:



```
>> demoo  
  
A =  
  
    2.5000  
  
B =  
  
    3  
  
fx >>
```

[New to MATLAB? See Help](#)

```
>> demo0  
  
A =  
  
    2.5000  
  
B =  
  
    3  
  
fx >>  
  
>> demo0  
  
A =  
  
    2.5000  
  
B =  
  
    2  
  
fx >>
```

7. Write a program to demonstrate the use of functions length(), ndims(), numel(), and size(). How they are different for 1-D and 2-D array? How many functions are having same values w.r.t 1-D and 2-D array.

Explanation:

ndims() returns the number of dimensions in the array A. The number of dimensions is always greater than or equal to 2. The function ignores trailing singleton dimensions, for which size(A,dim) = 1.

length() returns the length of the largest array dimension in X

size () returns a row vector whose elements are the lengths of the corresponding dimensions of A

numel() returns the number of elements, n, in array A, equivalent to prod(size())

Code:

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 -     A = 1:5;
2 -     B = size(A)
```

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 -     A = 1:5;
2 -     B = length(A)
```

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 -     A = 1:5;
2 -     B = ndims(A)
```

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 -     A = 1:5;
2 -     B = numel(A)
```

Output:

```
>> demoo
B =
1      5
>>
```

```
>> demoo  
B =  
5  
fx >>
```

```
>> demoo  
B =  
2  
fx >>
```

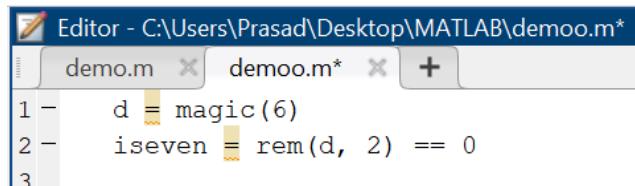
```
>> demoo  
B =  
5  
fx >>
```

4. Write a program to create a magic array of size 6 and check whether each element is odd or even and display the output accordingly.

Explanation:

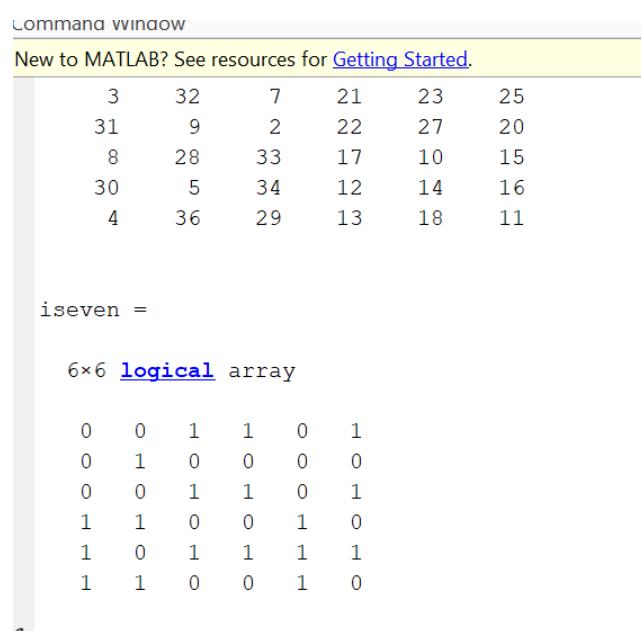
First create a magic array of size 6 and than use iseven function to point out even numbers as 1 and odd as 0.

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m*
demo.m    demoo.m*    +
1 - d = magic(6)
2 - iseven = rem(d, 2) == 0
3
```

Output:



```
Command window
New to MATLAB? See resources for Getting Started.
   3   32    7   21   23   25
   31    9    2   22   27   20
    8   28   33   17   10   15
   30    5   34   12   14   16
    4   36   29   13   18   11

iseven =
 6×6 logical array

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

5. What is the difference between transpose (), reshape () and rot90 ()? Explain with program.

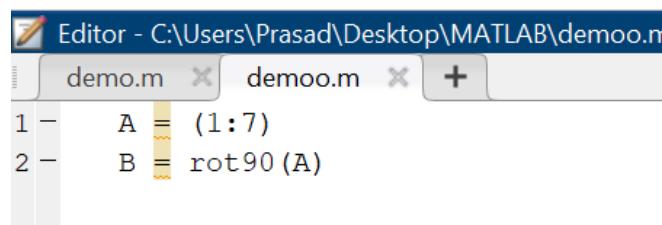
Explanation:

rot90() rotates array A counterclockwise by 90 degrees.

Transpose () = $B = A.'$ returns the nonconjugate transpose of A, that is, interchanges the row and column index for each element. If A contains complex elements, then $A.'$ does not affect the sign of the imaginary parts.

$B = \text{reshape}(A, \text{sz})$ reshapes A using the size vector, sz, to define size(B). For example, $\text{reshape}(A, [2,3])$ reshapes A into a 2-by-3 matrix. sz must contain at least 2 elements, and $\text{prod}(\text{sz})$ must be the same as $\text{numel}(A)$.

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demo.m
demo.m x demoo.m +
```

```
1 - A = (1:7)
2 - B = rot90(A)
```

Output:

```
>> demoo

A =
    1     2     3     4     5     6     7

B =
    7
    6
    5
    4
    3
    2
    1

fx ..
```

Code:

The screenshot shows the MATLAB Editor window titled "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". It contains two files: "demo.m" and "demoo.m". The "demo.m" file has the following code:

```
1 - A = magic(6)
2 - B = transpose(A)
```

Output:

The screenshot shows the MATLAB Command Window displaying the output of the code. It first shows the matrix A, which is a 6x6 magic square:

35	1	6	26	19	24
3	32	7	21	23	25
31	9	2	22	27	20
8	28	33	17	10	15
30	5	34	12	14	16
4	36	29	13	18	11

Then it shows the matrix B, which is the transpose of matrix A:

35	3	31	8	30	4
1	32	9	28	5	36
6	7	2	33	34	29
26	21	22	17	12	13
19	23	27	10	14	18
24	25	20	15	16	11

Code:

```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 -      A = 1:10
2 -      C = reshape(A, [2,5])
3 -      B = reshape(A, [5,2])
```

Command Window

Output:

```
Command window
New to MATLAB? See resources for Getting Started.
A =
1     2     3     4     5     6     7     8     9     10
C =
1     3     5     7     9
2     4     6     8    10
B =
1     6
2     7
3     8
4     9
```

6. Write a program to demonstrate fliplr () and flipud () in both 1-D and 2-D array.

fliplr() returns A with its columns flipped in the left-right direction (that is, about a vertical axis).

flipud () function to flip arrays in the vertical direction

Code:

fliplr() :

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m    demoo.m    +
```

```
1 - A = (1:10)
2 - B = fliplr(A)
```

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m    demoo.m    +
```

```
1 - A = cat(2, [1 2; 3 4], [5 6; 7 8])
2 - B = fliplr(A)
```

flipud ():

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m*

```
demo.m    demoo.m*    +
```

```
1 - A = (1:10)'
2 - B = flipud(A)
```

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m    demoo.m    +
```

```
1 - A = (1:10)
2 - B = flipud(A)
```

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

```
demo.m    demoo.m    +
```

```
1 - A = cat(2, [1 2; 3 4], [5 6; 7 8])
2 - B = flipud(A)
```

Output:

```
>> demo0

A =
1 2 3 4 5 6 7 8 9 10

B =
10 9 8 7 6 5 4 3 2 1

fx >>

NEW TO MATLAB? SEE RESULT
7
8
9
10

B =
10
9
8
7
6
5
4
3
2
1

fx . .

>> demo0

A =
1 2 3 4 5 6 7 8 9 10

B =
1 2 3 4 5 6 7 8 9 10

fx >>

A =
1 2 5 6
3 4 7 8

B =
3 4 7 8
1 2 5 6

fx >
```

A =

1	2	5	6
3	4	7	8

B =

6	5	2	1
8	7	4	3

fx >>

7. Write a program to demonstrate cumprod (). Display row-wise and column-wise.

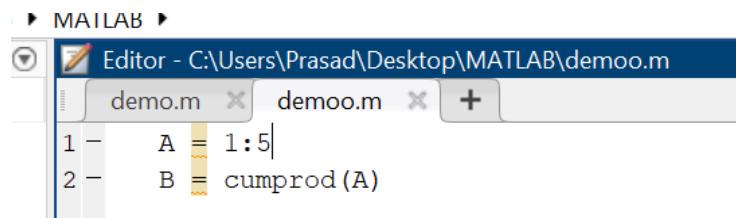
Explanation:

$B = \text{cumprod}(A)$ returns the cumulative product of A starting at the beginning of the first array dimension in A whose size does not equal 1.

- If A is a vector, then $\text{cumprod}(A)$ returns a vector containing the cumulative product of the elements of A .
- If A is a matrix, then $\text{cumprod}(A)$ returns a matrix containing the cumulative products for each column of A .
- If A is a multidimensional array, then $\text{cumprod}(A)$ acts along the first nonsingleton dimension.

i) 1-D

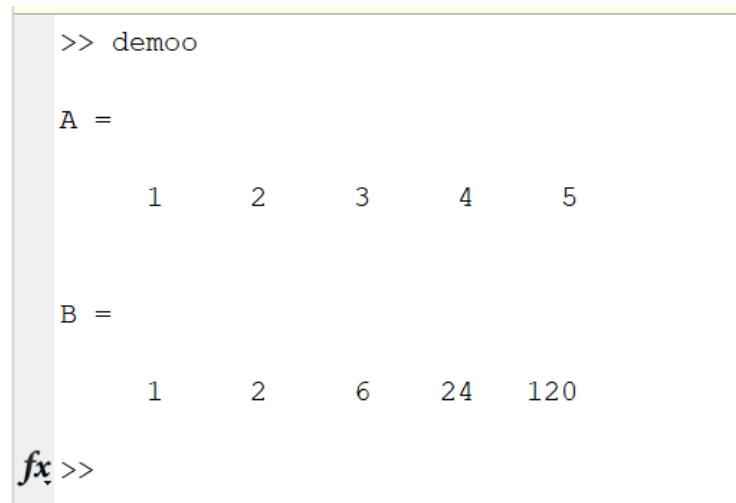
Code:



The screenshot shows the MATLAB Editor window. The title bar says "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". There are two tabs: "demo.m" and "demoo.m". The code in "demo.m" is:

```
1 - A = 1:5
2 - B = cumprod(A)
```

Output:

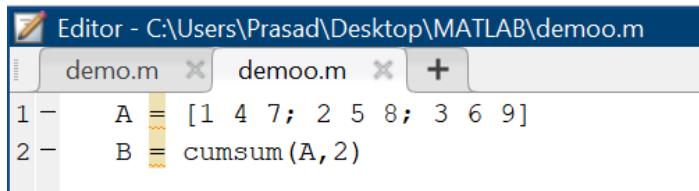


The screenshot shows the MATLAB Command Window. The input is ">> demoo". The output is:

```
>> demoo
A =
1     2     3     4     5
B =
1     2      6     24    120
fx >>
```

ii) 2-D

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
```

```
1 -      A = [1 4 7; 2 5 8; 3 6 9]
2 -      B = cumsum(A, 2)
```

Output:

```
>> demoo

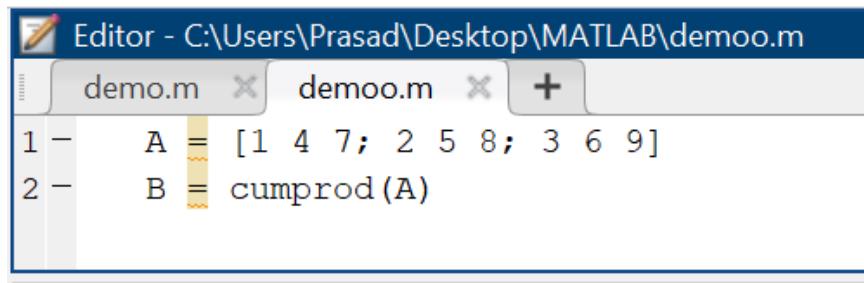
A =
1     4     7
2     5     8
3     6     9

B =
1     5     12
2     7     15
3     9     18

fx >>
```

iii) 3-D

Code:



The screenshot shows the MATLAB Editor window titled 'Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m'. It contains two tabs: 'demo.m' and 'demoo.m'. The 'demoo.m' tab is active and displays the following code:

```
1 - A = [1 4 7; 2 5 8; 3 6 9]
2 - B = cumprod(A)
```

Output:

```
>> demoo

A =
    1     4     7
    2     5     8
    3     6     9

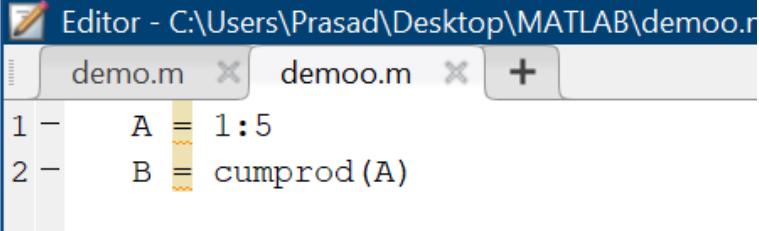
B =
    1     4     7
    2    20    56
    6   120   504

fx >>
```

8. Write a program to demonstrate reverse cumulative product. Display row-wise and column-wise.

i. 1-D

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m x demoo.m x +
```

```
1 - A = 1:5
2 - B = cumprod(A)
```

Output:

```
>> demoo

A =

    1     2     3     4     5

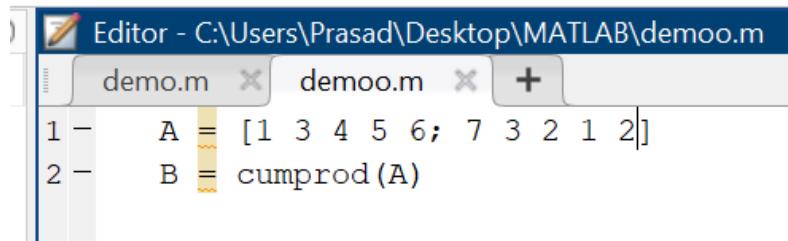
B =

    1     2      6     24    120

fx >>
```

ii. 2-D

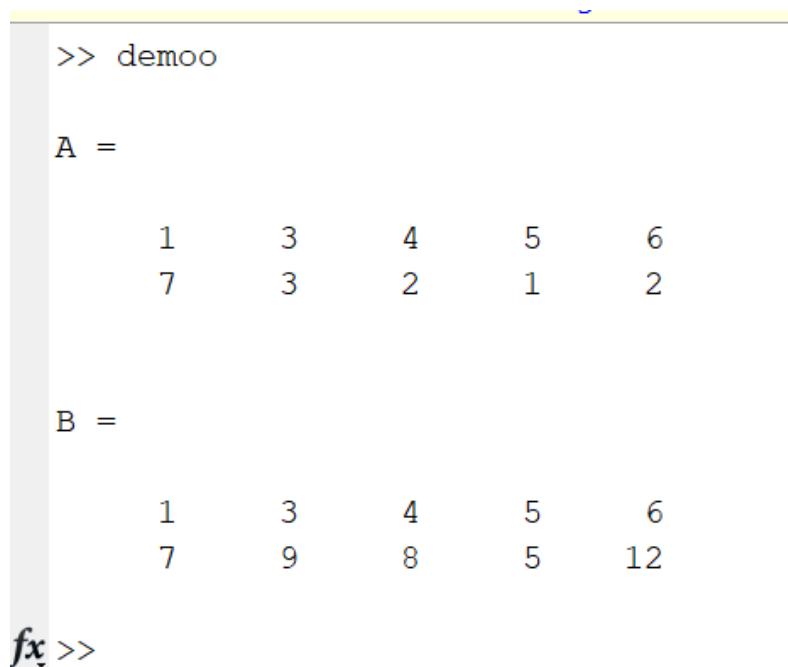
Code:



The screenshot shows the MATLAB Editor window with two files open: 'demo.m' and 'demoo.m'. The current file is 'demoo.m'. The code in 'demoo.m' is as follows:

```
1 - A = [1 3 4 5 6; 7 3 2 1 2]
2 - B = cumprod(A)
```

Output:



The screenshot shows the MATLAB Command Window with the following output:

```
>> demoo

A =
    1     3     4     5     6
    7     3     2     1     2

B =
    1     3     4     5     6
    7     9     8     5    12

fx >>
```

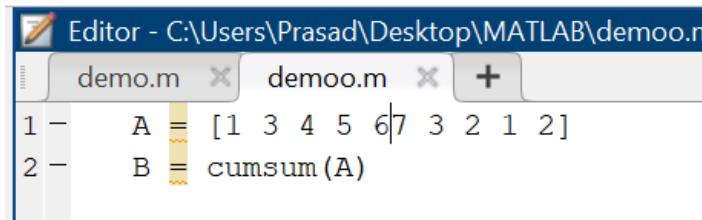
9. Write a program to demonstrate cumsum (). Display row-wise and column-wise.

$B = \text{cumsum}(A)$ returns the cumulative sum of A starting at the beginning of the first array dimension in A whose size does not equal 1. The output B has the same size as A.

- If A is a vector, then $\text{cumsum}(A)$ returns a vector containing the cumulative sum of the elements of A.
- If A is a matrix, then $\text{cumsum}(A)$ returns a matrix containing the cumulative sums of each column of A.
- If A is a multidimensional array, then $\text{cumsum}(A)$ acts along the first nonsingleton dimension.

i. 1-D

Code:



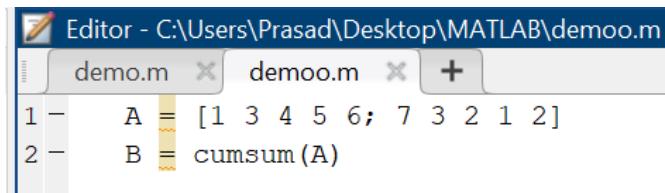
The screenshot shows the MATLAB Editor window titled "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". It contains two lines of code: "A = [1 3 4 5 6 7 3 2 1 2]" and "B = cumsum(A)". The second line is highlighted with a yellow selection bar.

Output:

```
>> demoo
A =
    1     3     4     5     6    7     3     2     1     2
B =
    1     4     8    13    80    83    85    86    88
fx >>
```

ii. 2-D

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
```

```
1 -      A = [1 3 4 5 6; 7 3 2 1 2]
2 -      B = cumsum(A)
```

Output:

```
>> demoo

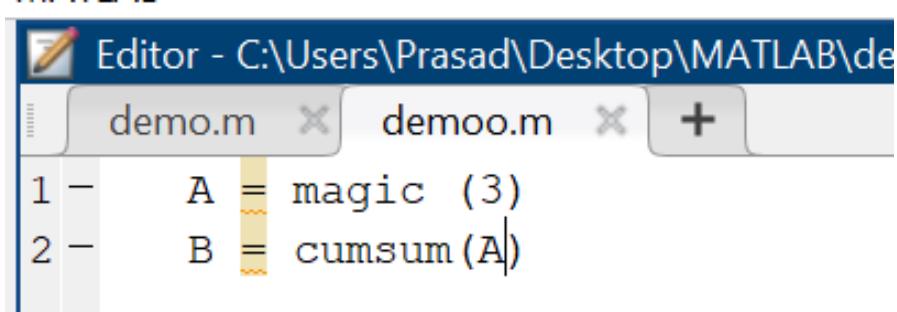
A =
1     3     4     5     6
7     3     2     1     2

B =
1     3     4     5     6
8     6     6     6     8

fx >>
```

iii. 3-D

Code:

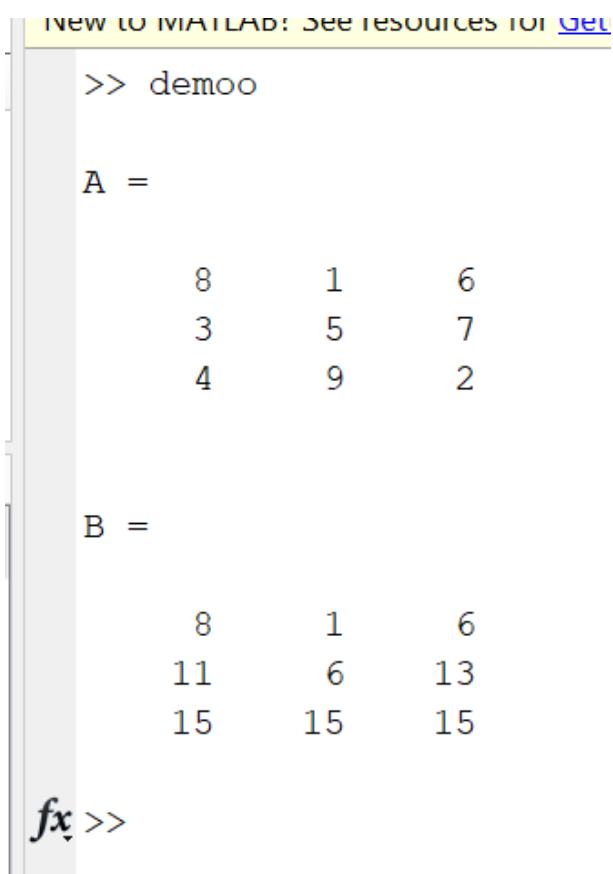


Editor - C:\Users\Prasad\Desktop\MATLAB\de

demo.m demoo.m +

1 - A = magic (3)
2 - B = cumsum (A)

Output:



New to MATLAB? See resources for [Get Started](#)

```
>> demoo

A =
8 1 6
3 5 7
4 9 2

B =
8 1 6
11 6 13
15 15 15

fx >>
```

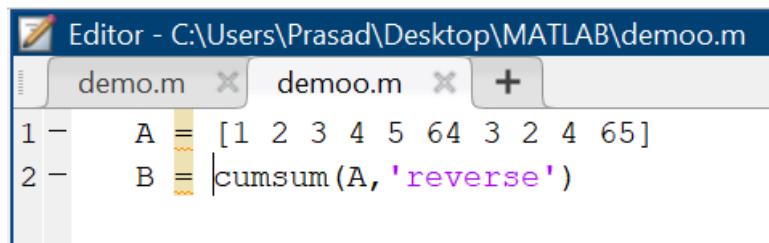
10. Write a program to demonstrate reverse cumulative sum. Display row-wise and column-wise.

Explanation:

$B = \text{cumsum}(_, \text{direction})$ specifies the direction using any of the previous syntaxes. For instance, $\text{cumsum}(A, 2, \text{'reverse'})$ returns the cumulative sum within the rows of A by working from end to beginning of the second dimension.

i. 1-D

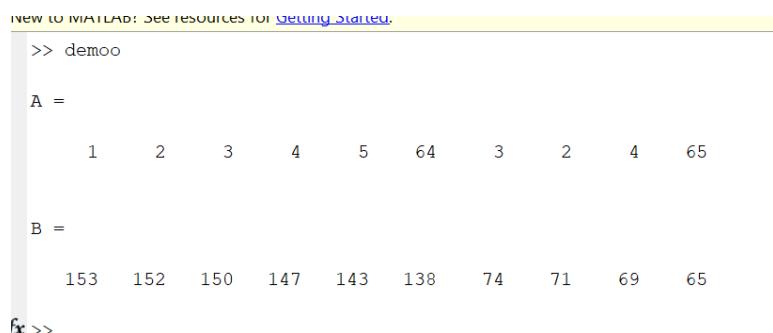
Code:



The screenshot shows the MATLAB Editor window with two files open: 'demo.m' and 'demoo.m'. The 'demoo.m' file is active and contains the following code:

```
1 - A = [1 2 3 4 5 64 3 2 4 65]
2 - B = cumsum(A, 'reverse')
```

Output:

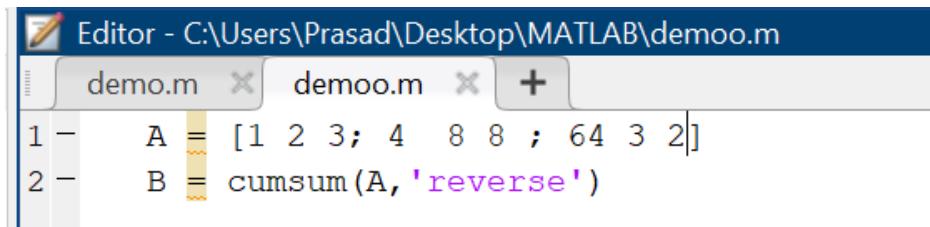


The screenshot shows the MATLAB Command Window with the following output:

```
>> demoo
A =
    1     2     3     4     5     64     3     2     4     65
B =
   153    152    150    147    143    138    74    71    69    65
>>
```

ii. 2-D

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 -      A = [1 2 3; 4 8 8 ; 64 3 2]
2 -      B = cumsum(A, 'reverse')
```

Output:

```
>> demoo

A =
1     2     3
4     8     8
64    3     2

B =
69     13     13
68     11     10
64      3      2

; >>
```

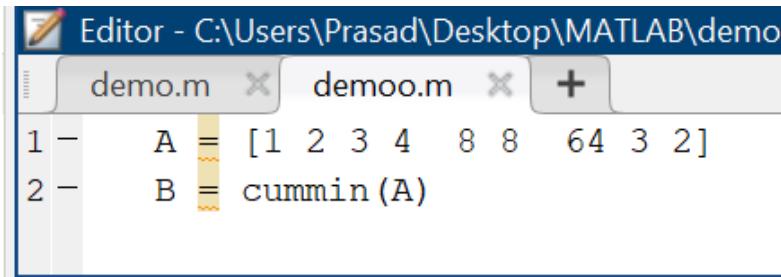
11. Write a program to demonstrate cummin (). Display row-wise and column-wise.

M = cummin(A) returns the cumulative minimum elements of A. By default, cummin(A) operates along the first array dimension whose size does not equal 1.

- If A is a vector, then cummin(A) returns a vector of the same size containing the cumulative minima of A.
- If A is a matrix, then cummin(A) returns a matrix of the same size containing the cumulative minima in each column of A.
- If A is a multidimensional array, then cummin(A) returns an array of the same size containing the cumulative minima along the first array dimension of A whose size does not equal 1.

i. 1-D

Code:



The screenshot shows the MATLAB Editor window titled "Editor - C:\Users\Prasad\Desktop\MATLAB\demo". It contains two files: "demo.m" and "demoo.m". The "demo.m" file is open and contains the following code:

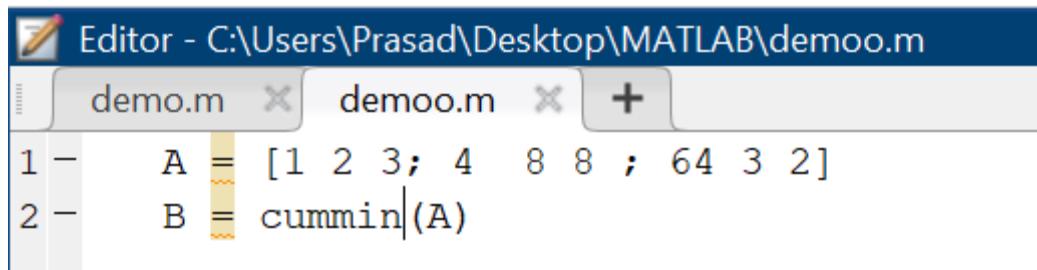
```
1 - A = [1 2 3 4 8 8 64 3 2]
2 - B = cummin(A)
```

Output:

```
>> demoo
A =
    1     2     3     4     8     8     64     3     2
B =
    1     1     1     1     1     1     1     1
fx >>
```

ii. 2-D

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
```

```
1 - A = [1 2 3; 4 8 8 ; 64 3 2]
2 - B = cummin(A)
```

Output:

```
>> demoo

A =
1 2 3
4 8 8
64 3 2

B =
1 2 3
1 2 3
1 2 2

x >>
```

12. Write a program to demonstrate reverse cumulative minimum. Display row-wise and column-wise.

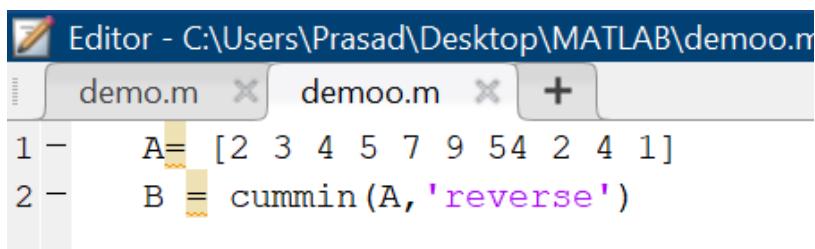
Explanation:

'reverse' works from end to 1 of the active dimension.

$M = \text{cummin}(_, \text{direction})$ optionally specifies the direction using any of the previous syntaxes. You must specify A and, optionally, can specify dim. For instance, $\text{cummin}(A, 2, \text{'reverse'})$ returns the cumulative minima of A by working from end to beginning of the second dimension of A.

i. 1-D

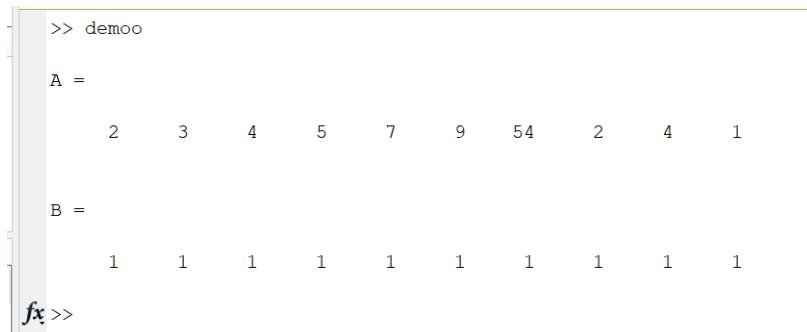
Code:



The screenshot shows the MATLAB Editor window with two files open: 'demo.m' and 'demoo.m'. The current file is 'demoo.m'. The code in 'demoo.m' is:

```
1 - A = [2 3 4 5 7 9 54 2 4 1]
2 - B = cummin(A, 'reverse')
```

Output:

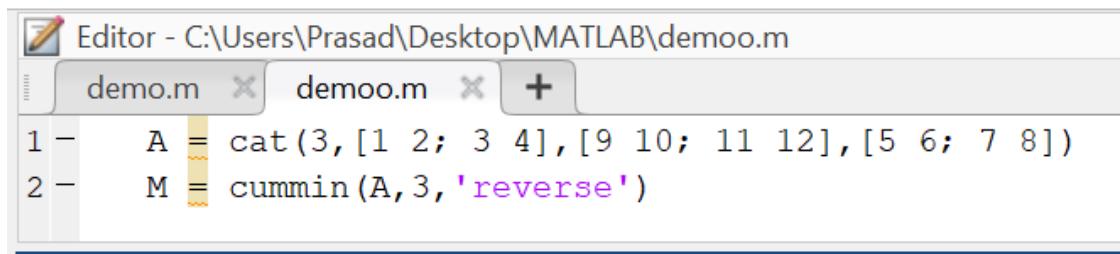


The screenshot shows the MATLAB Command Window with the following output:

```
>> demoo
A =
     2     3     4     5     7     9    54     2     4     1
B =
     1     1     1     1     1     1     1     1     1
fx >>
```

ii. 2-D

Code:



```
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m    demoo.m    +
1 -      A = cat(3,[1 2; 3 4],[9 10; 11 12],[5 6; 7 8])
2 -      M = cummin(A,3,'reverse')
```

Output:

```
>> demoo

A(:,:,1) =
1     2
3     4

A(:,:,2) =
9     10
11    12

A(:,:,3) =
5     6
7     8

fx

M(:,:,1) =
1     2
3     4

M(:,:,2) =
5     6
7     8

M(:,:,3) =
5     6
7     8
```

13. Write a program to demonstrate cummax (). Display row-wise and column-wise.

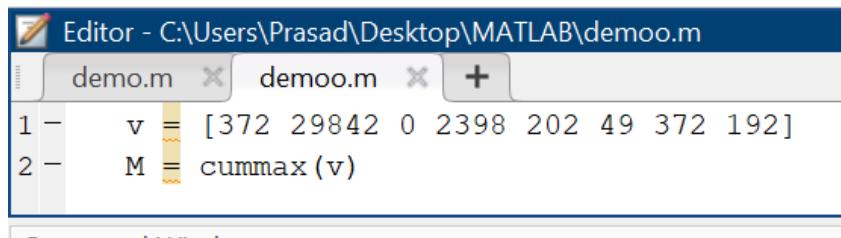
Explanation:

$M = \text{cummax}(A)$ returns the cumulative maximum elements of A . By default, $\text{cummax}(A)$ operates along the first array dimension whose size does not equal 1.

- If A is a vector, then $\text{cummax}(A)$ returns a vector of the same size containing the cumulative maxima of A .
- If A is a matrix, then $\text{cummax}(A)$ returns a matrix of the same size containing the cumulative maxima in each column of A .
- If A is a multidimensional array, then $\text{cummax}(A)$ returns an array of the same size containing the cumulative maxima along the first array dimension of A whose size does not equal 1.

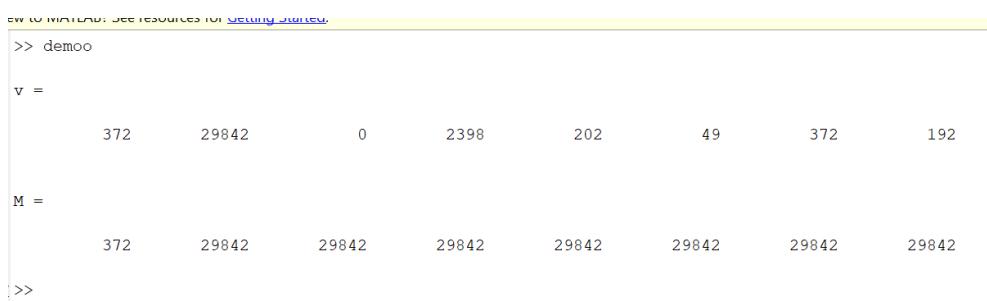
i. 1-D

Code:



The screenshot shows the MATLAB Editor window titled "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". It contains two lines of code: "v = [372 29842 0 2398 202 49 372 192]" and "M = cummax(v)". The variable "v" is highlighted in yellow.

Output:

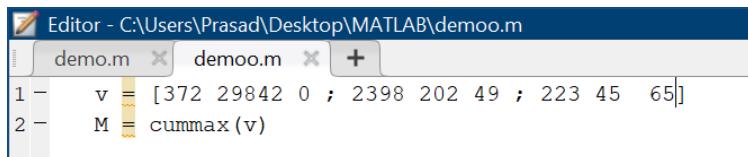


The screenshot shows the MATLAB Command Window with the following output:

```
>> demoo
v =
    372     29842         0     2398     202      49     372     192
M =
    372     29842     29842     29842     29842     29842     29842
>>
```

ii. 2-D

Code:



The screenshot shows the MATLAB Editor window titled 'Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m'. It contains two lines of code: 'v = [372 29842 0 ; 2398 202 49 ; 223 45 65]' and 'M = cummax(v)'. The variable 'v' is highlighted in yellow.

Output:

```
>> demoo

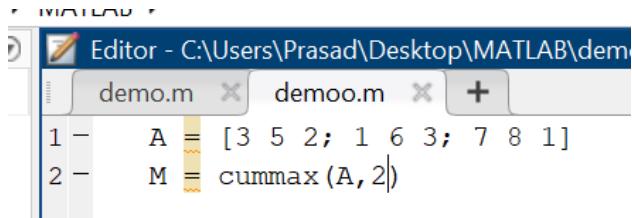
v =
    372      29842      0
    2398     202       49
    223      45        65

M =
    372      29842      0
    2398     29842      49
    2398     29842      65

>>
```

iii. 3-D

Code:



Editor - C:\Users\Prasad\Desktop\MATLAB\dem
demo.m demoo.m +
1 - A = [3 5 2; 1 6 3; 7 8 1]
2 - M = cummax(A, 2)

Output:

```
>> demoo  
  
A =  
  
    3     5     2  
    1     6     3  
    7     8     1  
  
M =  
  
    3     5     5  
    1     6     6  
    7     8     8  
  
x >>
```

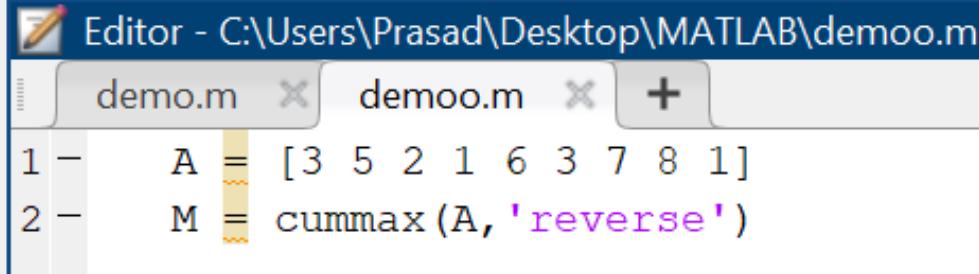
14. Write a program to demonstrate reverse cumulative maximum. Display row-wise and column-wise.

Explanation:

$M = \text{cummax}(_, \text{direction})$ optionally specifies the direction using any of the previous syntaxes. You must specify A and, optionally, can specify dim. For instance, $\text{cummax}(A, 2, \text{'reverse'})$ returns the cumulative maxima of A by working from end to beginning of the second dimension of A.

i. 1-D

Code:



The screenshot shows the MATLAB Editor window with two tabs: 'demo.m' and 'demoo.m'. The 'demoo.m' tab is active and contains the following code:

```
1 - A = [3 5 2 1 6 3 7 8 1]
2 - M = cummax(A, 'reverse')
```

Output:

```
>> demoo

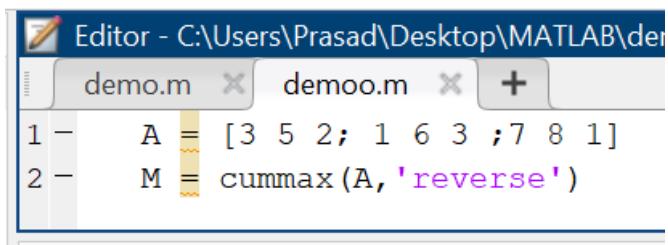
A =
    3     5     2     1     6     3     7     8     1

M =
    8     8     8     8     8     8     8     8     1

>>
```

ii. 2-D

Code:



The screenshot shows the MATLAB Editor window titled 'Editor - C:\Users\Prasad\Desktop\MATLAB\dem'. It contains two files: 'demo.m' and 'demoo.m'. The current file is 'demoo.m', which contains the following code:

```
1 - A = [3 5 2; 1 6 3 ;7 8 1]
2 - M = cummax(A, 'reverse')
```

Output:

```
>> demoo

A =
    3     5     2
    1     6     3
    7     8     1

M =
    7     8     3
    7     8     3
    7     8     1

fx >>
```

15. What is the difference between sort () and sortrows ()? Explain with a program.

Explanation:

The sortrows() function provides additional flexibility for subsorting over multiple columns of matrix or table inputs.

The sort () function and the relational operators use different orderings for complex numbers.

Sort()

Code:

▶ MATLAB ▶

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m x demoo.m x +

```
1 - A = [34 78 -7 25 51 78 -19 34 12];
2 - B = sort(A)
```

Output:

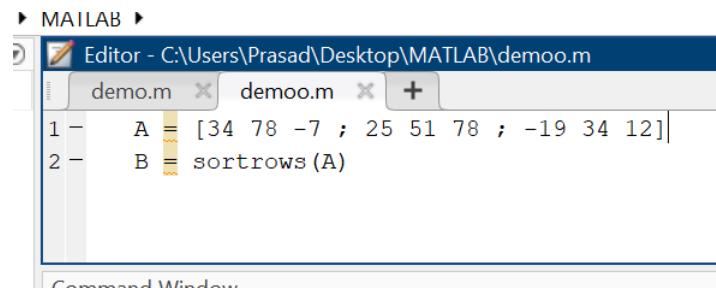
Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> demoo
B =
-19    -7    12    25    34    34    51    78    78
fx >>
```

Sortrows()

Code:



Output:

The screenshot shows the MATLAB Command Window. It displays the output of the 'demoo' script. The variable 'A' is defined as a 3x3 matrix with elements 34, 78, -7; 25, 51, 78; -19, 34, 12. The variable 'B' is defined as a 3x3 matrix where the rows of 'A' are sorted in ascending order: -19, 34, 12; 25, 51, 78; 34, 78, -7. The command 'fx >>' is shown at the bottom.

```
>> demoo

A =
    34    78     -7
    25    51     78
   -19    34     12

B =
   -19    34     12
    25    51     78
    34    78     -7

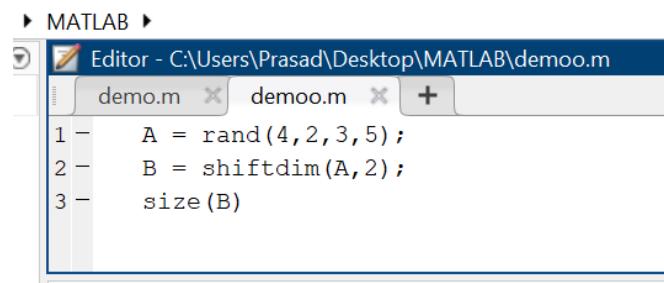
fx >>
```

16. Write a program to demonstrate the shiftdim () function.

Explanation:

So I have created a 4-by-2-by-3-by-5 array and I am just shifting the dimensions 2 positions to the left, wrapping the first 2 lengths to the last 2 dimensions.

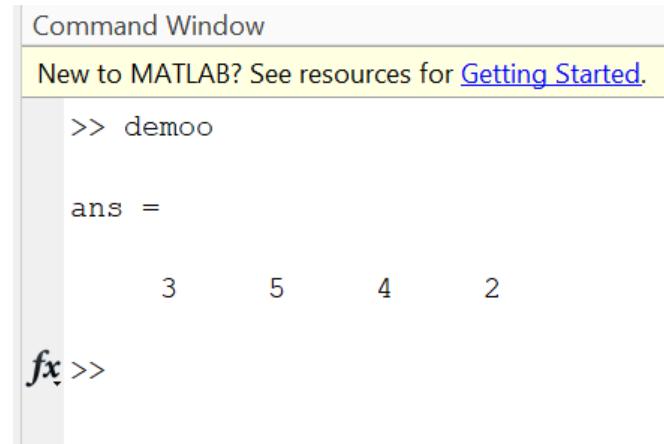
Code:



The screenshot shows the MATLAB Editor window. The title bar says "Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m". There are two tabs: "demo.m" and "demoo.m". The code area contains the following MATLAB script:

```
1 - A = rand(4,2,3,5);
2 - B = shiftdim(A,2);
3 - size(B)
```

Output:



The screenshot shows the MATLAB Command Window. The title bar says "Command Window". A yellow banner at the top says "New to MATLAB? See resources for [Getting Started](#)". The command line shows the execution of the script:

```
>> demoo
ans =
    3     5     4     2
fx >>
```