

EXPERIMENT 3

Take any matrices as A, B and C.

1. Addition & Subtraction: $C = A + B$, $C = A - B$
2. Matrix Multiplication: $C = A * B$ (requires inner dimensions to match)
3. Element-wise Multiplication: $C = A .* B$
4. Division: $C = A / B$, $C = A ./ B$
5. Transpose: A'
6. Determinant: $\det(A)$
7. Inverse: $\text{inv}(A)$ Rank: $\text{rank}(A)$

```
% Define matrices
A = [1 2 3; 4 5 6; 7 8 9];
B = [9 8 7; 6 5 4; 3 2 1];
C = [2 4 6; 1 3 5; 7 9 11];

% --- 1. Addition ---
add_AB = A + B;

% --- 2. Subtraction ---
sub_AB = A - B;

% --- 3. Matrix Multiplication ---
mul_AB = A * B; % valid since both are 3x3

% --- 4. Element-wise Multiplication ---
ele_mul_AB = A .* B;

% --- 5. Matrix Division ---
div_AB = A / B; % equivalent to A * inv(B)

% --- 6. Element-wise Division ---
ele_div_AB = A ./ B;

% --- 7. Transpose ---
A_transpose = A';

% Display all results
disp('Addition:');
disp(add_AB);
disp('Subtraction:');
disp(sub_AB);
disp('Matrix Multiplication:');
disp(mul_AB);
disp('Element-wise Multiplication:');
disp(ele_mul_AB);
disp('Matrix Division:');
disp(div_AB);
disp('Element-wise Division:');
disp(ele_div_AB);
disp('Transpose of A:');
disp(A_transpose);
disp('Determinant of A:');
disp(A_determinant);

Warning: Matrix is singular to working precision.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
1.54197e-18.
```

```
% --- 7. Transpose ---
A_transpose = A';

% --- 8. Determinant ---
A_determinant = det(A);

% --- 9. Inverse ---
A_inverse = inv(A);

% --- 10. Rank ---
A_rank = rank(A);

% Display all results
disp('Addition:');
disp(add_AB);
disp('Subtraction:');
disp(sub_AB);
disp('Matrix Multiplication:');
disp(mul_AB);
disp('Element-wise Multiplication:');
disp(ele_mul_AB);
disp('Matrix Division:');
disp(div_AB);
disp('Element-wise Division:');
disp(ele_div_AB);
disp('Transpose of A:');
disp(A_transpose);
disp('Determinant of A:');
disp(A_determinant);
disp('Inverse of A:');
disp(A_inverse);
disp('Rank of A:');
disp(A_rank);

Transpose of A:
1 4 7
2 5 8
3 6 9

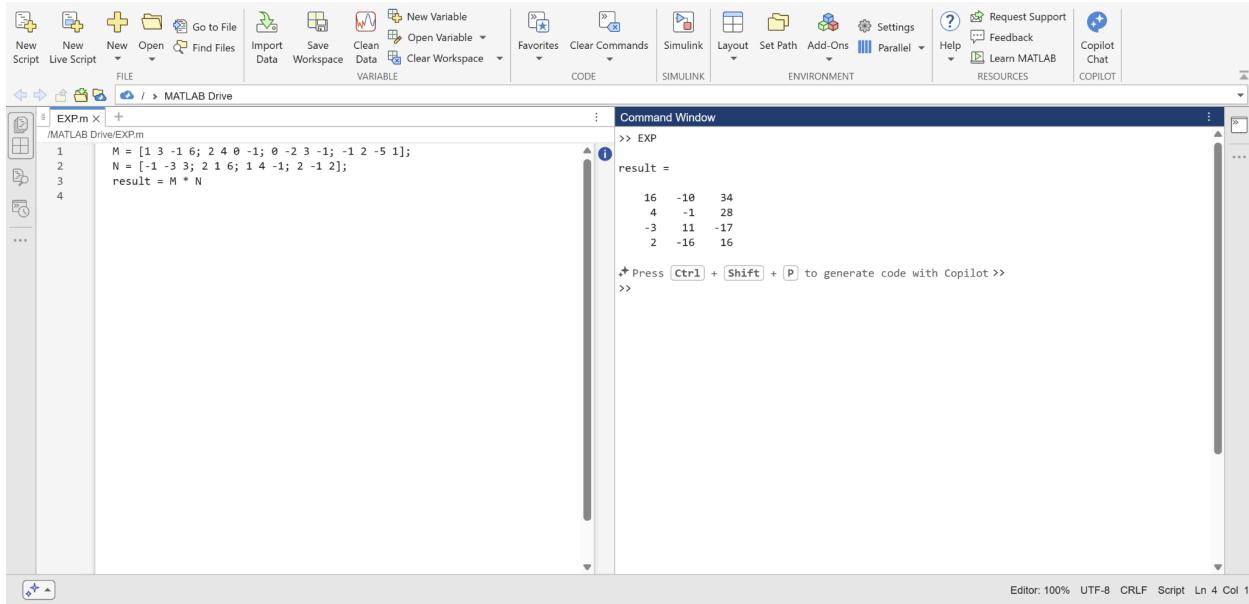
Determinant of A:
6.6613e-16

Inverse of A:
1.0e+16 *
-0.4504 0.9007 -0.4504
0.9007 -1.8014 0.9007
-0.4504 0.9007 -0.4504

Rank of A:
2
```

EXERCISE:

1. Enter the matrix $M = [1,3,-1,6;2,4,0,-1;0,-2,3,-1;-1,2,-5,1]$ and $N = [-1,-3,3;2,1,6;1,4,-1;2,-1,2]$, perform multiplication on M and N. Can order of multiplication be switched? Why or why not?



The screenshot shows the MATLAB interface with the Command Window active. The code entered is:

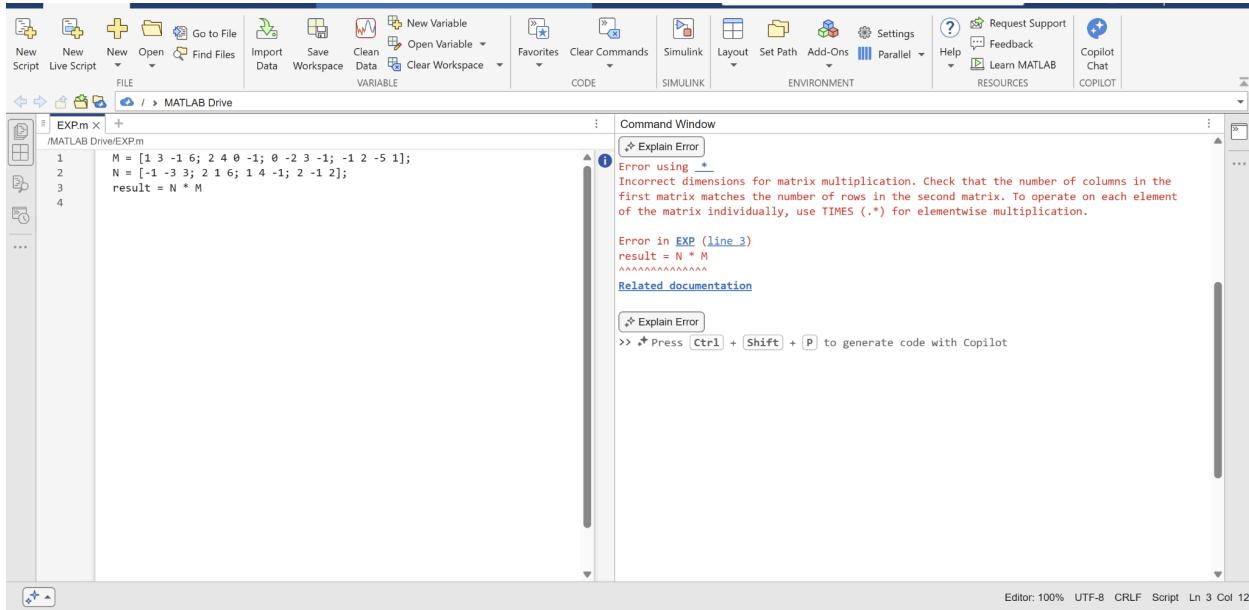
```
>> EXPm
/MATLAB Drive/EXPm
1 M = [1 3 -1 6; 2 4 0 -1; 0 -2 3 -1; -1 2 -5 1];
2 N = [-1 -3 3; 2 1 6; 1 4 -1; 2 -1 2];
3 result = M * N
4
```

The output shows the result of the multiplication:

```
result =
16   -10    34
 4    -1    28
 -3    11   -17
 2   -16    16
```

A note at the bottom of the Command Window says: "Press **Ctrl** + **Shift** + **P** to generate code with Copilot >>"

The order of multiplication cannot be switched because the dimensions are incompatible and, even when compatible, matrix multiplication is not commutative.



The screenshot shows the MATLAB interface with the Command Window active. The code entered is:

```
>> EXPm
/MATLAB Drive/EXPm
1 M = [1 3 -1 6; 2 4 0 -1; 0 -2 3 -1; -1 2 -5 1];
2 N = [-1 -3 3; 2 1 6; 1 4 -1; 2 -1 2];
3 result = N * M
4
```

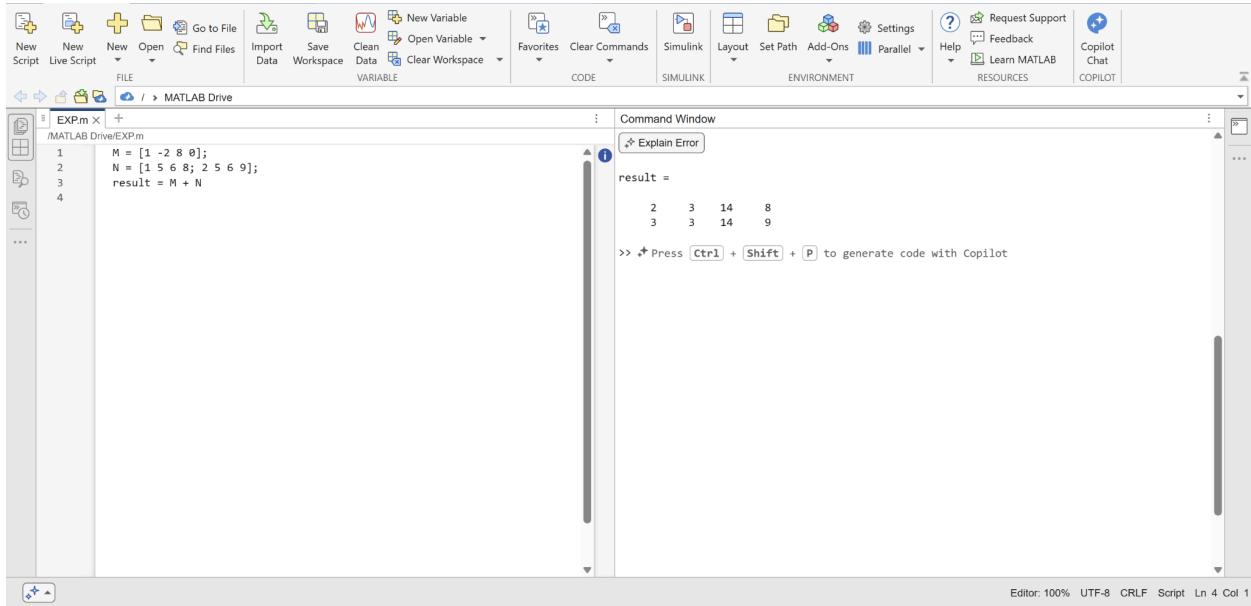
An error message is displayed in the Command Window:

```
Error using *
Incorrect dimensions for matrix multiplication. Check that the number of columns in the
first matrix matches the number of rows in the second matrix. To operate on each element
of the matrix individually, use TIMES (*) for elementwise multiplication.
```

Below the error message, it says "Error in EXP (line 3)" and "result = N * M". A "Related documentation" link is also visible.

A note at the bottom of the Command Window says: "Press **Ctrl** + **Shift** + **P** to generate code with Copilot >>"

2. Enter the matrix $M = [1, -2, 8, 0]$ and $N = [1 \ 5 \ 6 \ 8; 2 \ 5 \ 6 \ 9]$ Perform addition on M and N and see how matlab reacts.



The screenshot shows the MATLAB interface with the Command Window open. The code entered is:

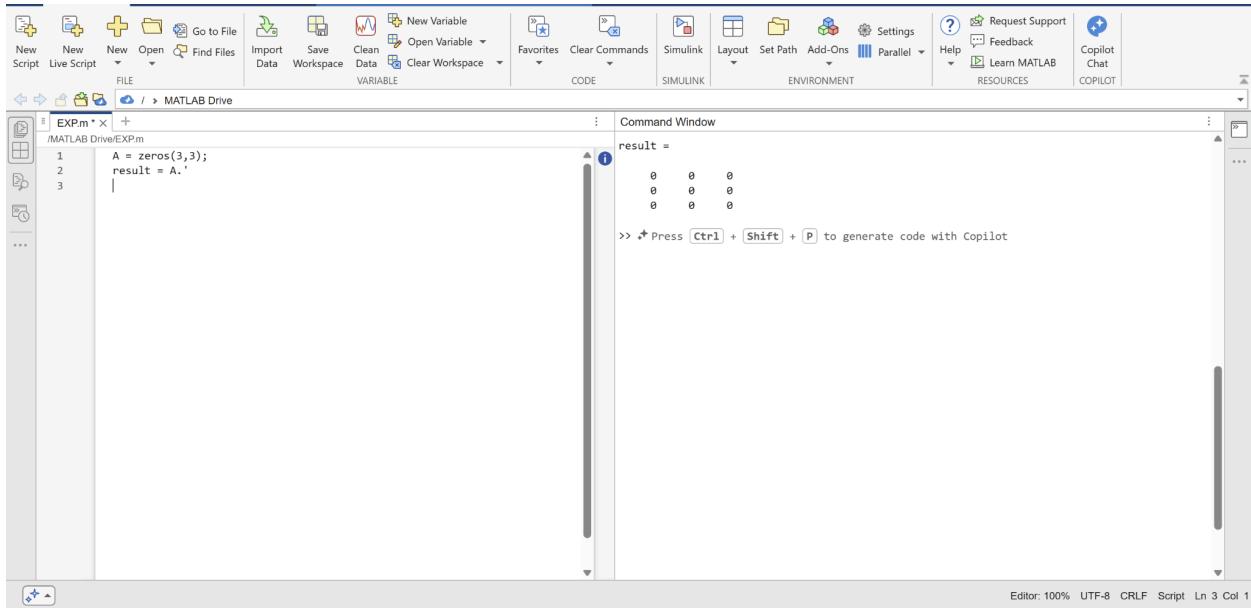
```
M = [1 -2 8 0];
N = [1 5 6 8; 2 5 6 9];
result = M + N
```

The output in the Command Window is:

```
result =
2     3    14     8
3     3    14     9
```

A tooltip "Press Ctrl + Shift + P to generate code with Copilot" is visible at the bottom of the window.

3. Find the transpose of null matrix using MATLAB SIMULINK.



The screenshot shows the MATLAB interface with the Command Window open. The code entered is:

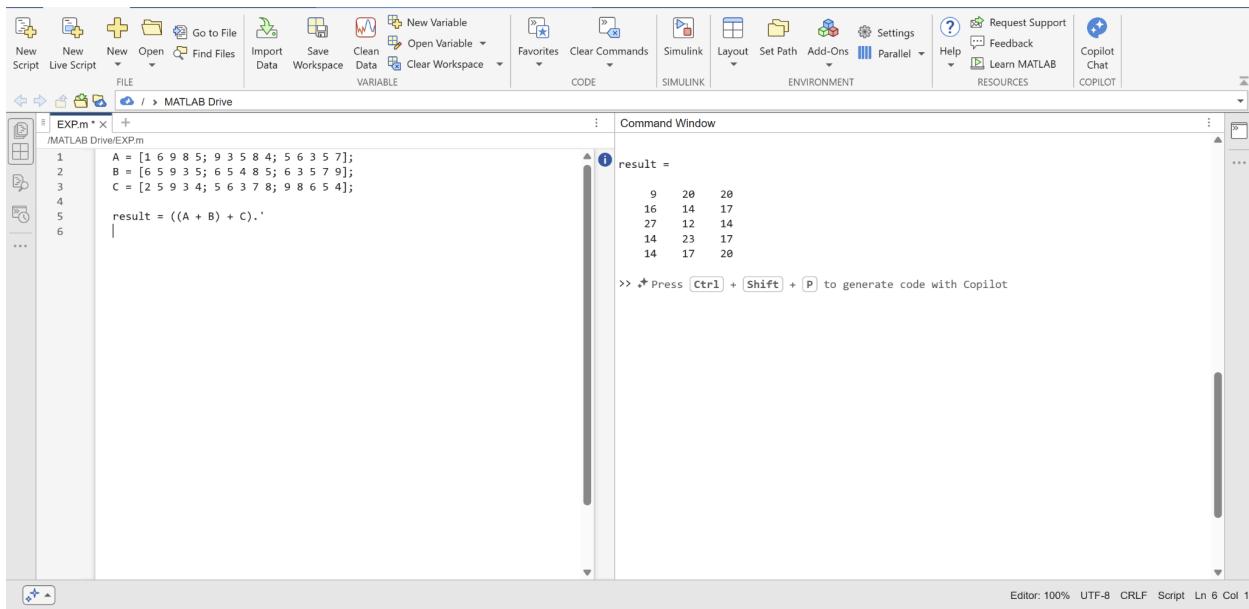
```
A = zeros(3,3);
result = A.'
```

The output in the Command Window is:

```
result =
0     0     0
0     0     0
0     0     0
```

A tooltip "Press Ctrl + Shift + P to generate code with Copilot" is visible at the bottom of the window.

4. Enter the matrix A = [1 6 9 8 5; 9 3 5 8 4; 5 6 3 5 7], B = [6 5 9 3 5; 6 5 4 8 5; 6 3 5 7 9], C = [2 5 9 3 4; 5 6 3 7 8; 9 8 6 5 4] Find [(A+B)+C]T using MATLAB SIMULINK.



The screenshot shows the MATLAB interface with the Command Window active. The code entered is:

```

EXP.m * X
/MATLAB Drive/EXPm
1 A = [1 6 9 8 5; 9 3 5 8 4; 5 6 3 5 7];
2 B = [6 5 9 3 5; 6 5 4 8 5; 6 3 5 7 9];
3 C = [2 5 9 3 4; 5 6 3 7 8; 9 8 6 5 4];
4 result = ((A + B) + C).';

```

The output displayed is:

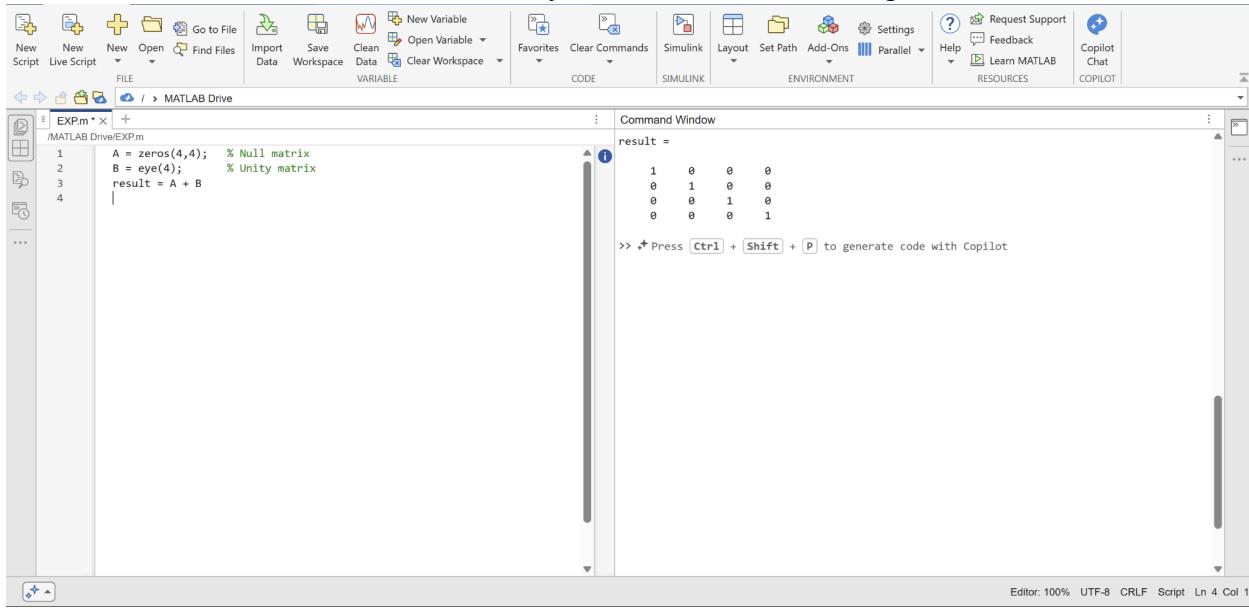
```

result =
9   20   20
16   14   17
27   12   14
14   23   17
14   17   20

```

A tooltip at the bottom right of the window says: "Press **Ctrl** + **Shift** + **P** to generate code with Copilot".

5. Find the addition of null matrix and unity matrix of order 4x4 using MATLAB SIMULINK.



The screenshot shows the MATLAB interface with the Command Window active. The code entered is:

```

EXP.m * X
/MATLAB Drive/EXPm
1 A = zeros(4,4); % Null matrix
2 B = eye(4); % Unity matrix
3 result = A + B

```

The output displayed is:

```

result =
1   0   0   0
0   1   0   0
0   0   1   0
0   0   0   1

```

A tooltip at the bottom right of the window says: "Press **Ctrl** + **Shift** + **P** to generate code with Copilot".

6. Find the multiplication of null matrix and unity matrix of order 3x3 using MATLAB SIMULINK.

The screenshot shows the MATLAB interface with the Command Window and Editor tabs active. The Editor tab displays a script named EXP.m containing the following code:

```
EXP.m % Null matrix
1 A = zeros(3,3); % Null matrix
2 B = eye(3); % Unity matrix
3 result = A * B
```

The Command Window shows the output of the code execution:

```
result =
0 0 0
0 0 0
0 0 0
```

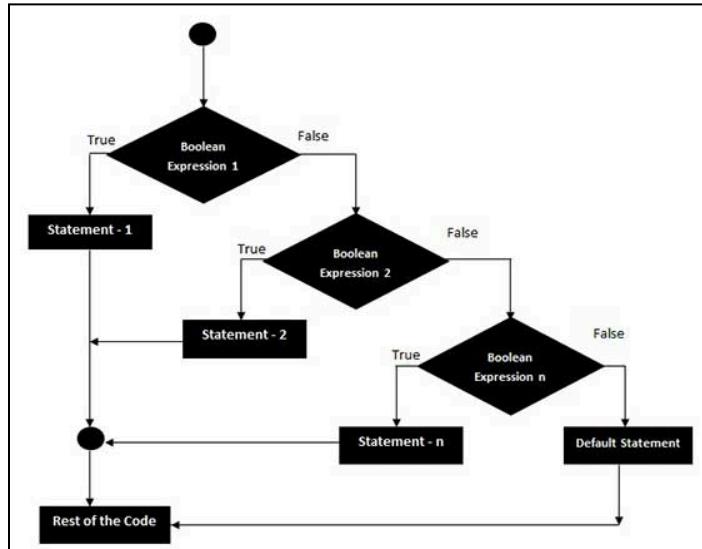
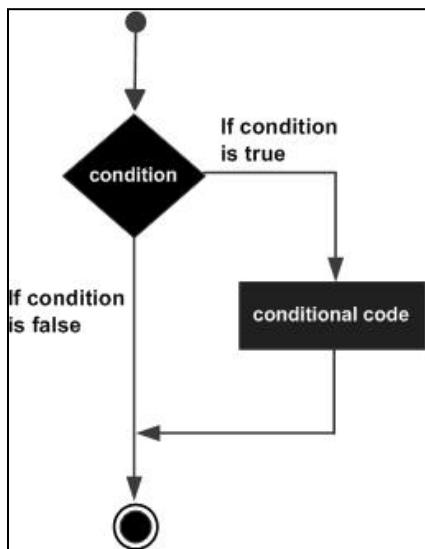
A tooltip message at the bottom right of the Command Window says: "Press Ctrl + Shift + P to generate code with Copilot".

At the bottom of the window, status bar text includes: "Editor: 100% UTF-8 CRLF Script Ln 4 Col 1".

EXPERIMENT 8

EXERCISE

1. Prepare functioning flowcharts for if-end, if-else-end, and if-elseif-else-end.



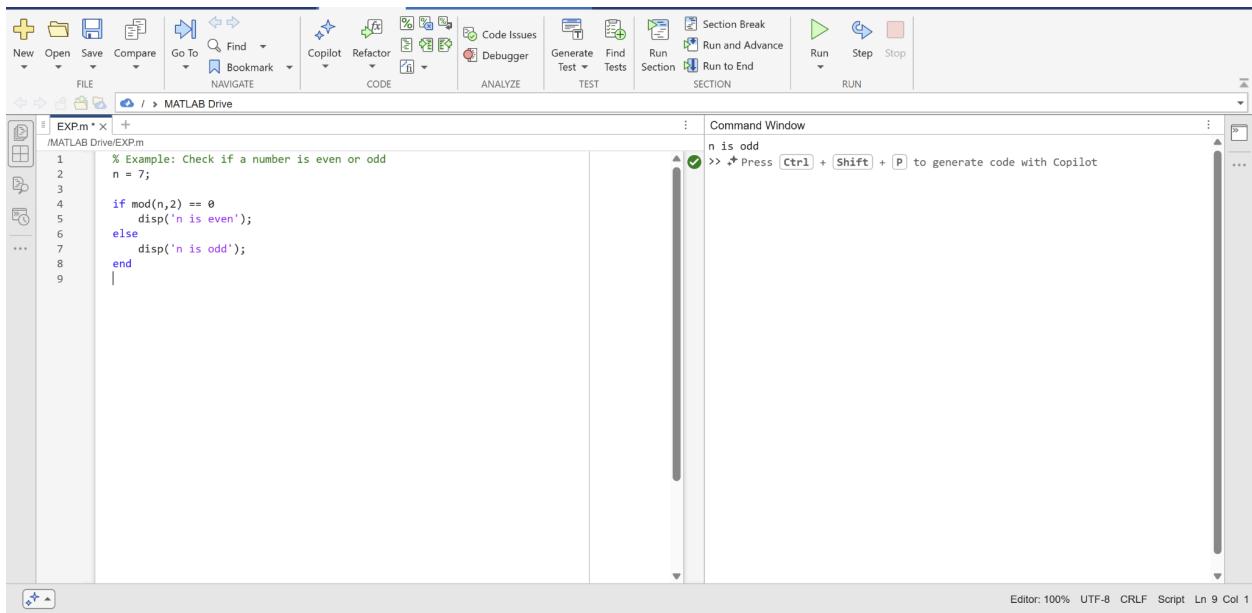
if-end Statement

A screenshot of the MATLAB IDE interface. The top menu bar includes FILE, CODE, TEST, SECTION, and RUN sections. The central workspace shows a script named EXP.m with the following code:

```
%if-end Statement
% Example: Check if a number is positive
x = 5;
if x > 0
    disp('x is positive');
end
```

The Command Window on the right displays the output: "x is positive". A status bar at the bottom indicates "Editor: 100% UTF-8 CRLF Script Ln 6 Col 27".

if–else–end Statement

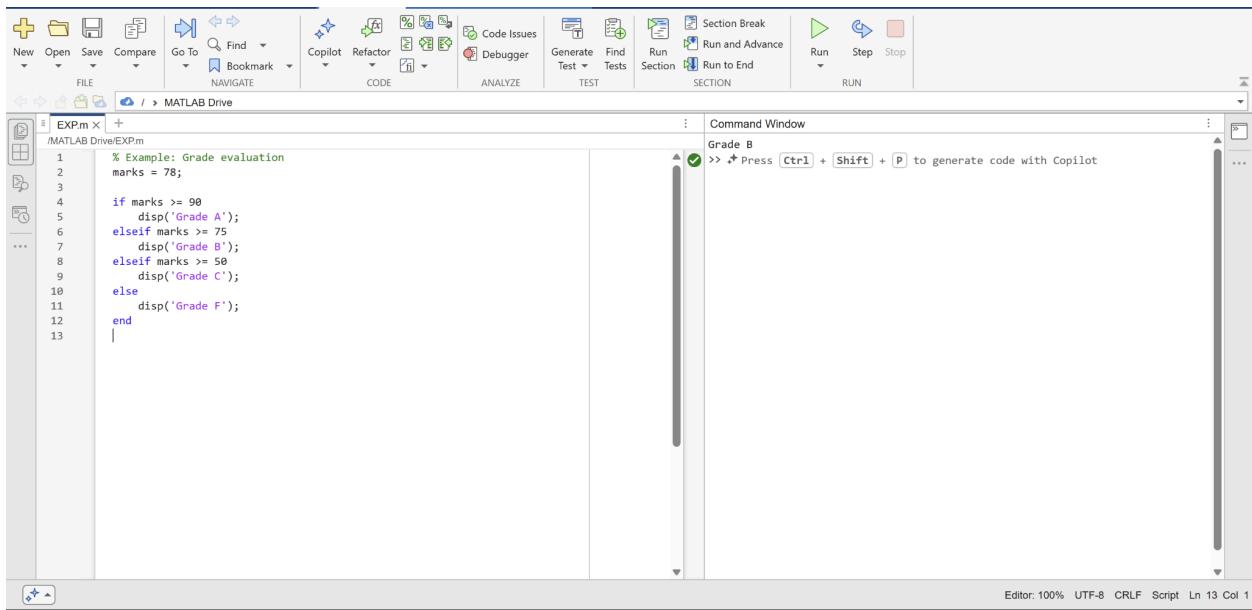


The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, CODE, ANALYZE, TEST, SECTION, and RUN. The left pane displays a script named EXPm.m with the following code:

```
1 % Example: Check if a number is even or odd
2 n = 7;
3
4 if mod(n,2) == 0
5     disp('n is even');
6 else
7     disp('n is odd');
8 end
9
```

The right pane is the Command Window, which shows the output: "n is odd". A status bar at the bottom indicates "Editor: 100% UTF-8 CRLF Script Ln 9 Col 1".

if–elseif–else–end Statement



The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, CODE, ANALYZE, TEST, SECTION, and RUN. The left pane displays a script named EXPm.m with the following code:

```
1 % Example: Grade evaluation
2 marks = 78;
3
4 if marks >= 90
5     disp('Grade A');
6 elseif marks >= 75
7     disp('Grade B');
8 elseif marks >= 50
9     disp('Grade C');
10 else
11     disp('Grade F');
12 end
13
```

The right pane is the Command Window, which shows the output: "Grade B". A status bar at the bottom indicates "Editor: 100% UTF-8 CRLF Script Ln 13 Col 1".

2. The following were the daily maximum temperatures (in F) in Washington, DC, during the month of April 2002: 58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64 74 63 69 (data from the U.S. National Oceanic and Atmospheric Administration). Use relational and logical operations to determine the following:

(a) The number of days the temperature was above 75 .

The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, NAVIGATE, CODE, ANALYZE, TEST, SECTION, and RUN. The left sidebar has icons for New, Open, Save, Compare, Go To, Find, Copilot, Refactor, Debugger, Generate Test, Find Tests, Run Section, Run and Advance, Run to End, Section Break, Run, Step, and Stop. The central workspace shows a script named EXP.m with the following code:

```
1 temp = [58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64];  
2 days_above_75 = sum(temp > 75);  
3 disp(['Number of days above 75 F: ', num2str(days_above_75)]);
```

The Command Window on the right displays the output: "Number of days above 75 F: 7". A tooltip at the bottom right says: "Press Ctrl + Shift + P to generate code with Copilot". The status bar at the bottom indicates: Editor: 100% UTF-8 CRLF Script Ln 4 Col 1.

(b) The number of days the temperature was between 65 and 80 .

The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, NAVIGATE, CODE, ANALYZE, TEST, SECTION, and RUN. The left sidebar has icons for New, Open, Save, Compare, Go To, Find, Copilot, Refactor, Debugger, Generate Test, Find Tests, Run Section, Run and Advance, Run to End, Section Break, Run, Step, and Stop. The central workspace shows a script named EXP.m with the following code:

```
1 temp = [58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64];  
2 %  
3 days_above_75 = sum(temp > 75);  
4 disp(['Number of days above 75 F: ', num2str(days_above_75)]);  
5  
6 %  
7 days_65_to_80 = sum(temp >= 65 & temp <= 80);  
8 disp(['Number of days between 65 and 80 F: ', num2str(days_65_to_80)]);  
9
```

The Command Window on the right displays three outputs: "Number of days above 75 F: 7", "Number of days above 75 F: 7", and "Number of days between 65 and 80 F: 12". A tooltip at the bottom right says: "Press Ctrl + Shift + P to generate code with Copilot". The status bar at the bottom indicates: Editor: 100% UTF-8 CRLF Script Ln 6 Col 3.

(c) The days of the month when the temperature was between 50 and 60 .

The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, NAVIGATE, CODE, ANALYZE, TEST, SECTION, and RUN. The left pane displays a script named EXP.m:

```

1 temp = [58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64];
2 %
3 days_above_75 = sum(temp > 75);
4 disp(['Number of days above 75 F: ', num2str(days_above_75)]);
5
6 %
7 days_65_to_80 = sum(temp >= 65 & temp <= 80);
8 disp(['Number of days between 65 and 80 F: ', num2str(days_65_to_80)]);
9
10 %
11 days_50_to_60 = find(temp >= 50 & temp <= 60);
12 disp('Days when temperature was between 50 and 60 F:');
13 disp(days_50_to_60);
14

```

The right pane is the Command Window, showing the results of the script execution:

```

Number of days above 75 F: 7
Number of days between 65 and 80 F: 12
Days when temperature was between 50 and 60 F:
1 4 5 7 21 23

```

Below the Command Window, a note says: "Press **Ctrl** + **Shift** + **P** to generate code with Copilot".

3. A worker is paid according to his hourly wage up to 40 hours, and 50%more for overtime. Write a program in a script file that calculates the pay to a worker. Theprogram asks the user to enter the number of hours and the hourly wage. Theprogram then displays the pay.

The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, NAVIGATE, CODE, ANALYZE, TEST, SECTION, and RUN. The left pane displays a script named EXP.m:

```

1 % Program to calculate worker's pay including overtime
2
3 hours = input('Enter the number of hours worked: ');
4 wage = input('Enter the hourly wage: ');
5
6 % Check if hours exceed 40 for overtime
7 if hours <= 40
8     pay = hours * wage;      % Regular pay
9 else
10    overtime_hours = hours - 40;
11    pay = 40 * wage + 1.5 * wage * overtime_hours; % Regular + Overtime pay
12 end
13
14 % Display the result
15 disp(['Total pay for the worker is: $', num2str(pay)]);
16

```

The right pane is the Command Window, showing the results of the script execution:

```

Enter the number of hours worked: 90
Enter the hourly wage: 50
Total pay for the worker is: $5750
>> |

```

Below the Command Window, a note says: "Press **Ctrl** + **Shift** + **P** to generate code with Copilot".

4. Write a program in a script file that creates an matrix with elements that have the following values. The value of each element in the first row is the number of the column. The value of each element in the first column is the number of the row. The rest of the elements each has a value equal to the sum of the element above it and the element to the left. When executed, the program asks the user to enter values for n and m.

The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, CODE, ANALYZE, TEST, and SECTION. Below the menu is a toolbar with icons for New, Open, Save, Compare, Go To, Find, Copilot, Refactor, Debugger, Generate Test, Run Section, Run and Advance, Run to End, Section Break, Run, Step, and Stop. The central workspace contains a script editor window titled 'EXPm' and a command window. The script code is as follows:

```

1 % Program to create a matrix based on specified pattern
2
3 % Ask user for matrix size
4 n = input('Enter the number of rows (n): ');
5 m = input('Enter the number of columns (m): ');
6
7 % Initialize matrix with zeros
8 A = zeros(n, m);
9
10 % Fill first row with column numbers
11 A(1, :) = 1:m;
12
13 % Fill first column with row numbers
14 A(:, 1) = 1:n;
15
16 % Fill remaining elements: sum of element above and element to the left
17 for i = 2:n
18     for j = 2:m
19         A(i,j) = A(i-1,j) + A(i,j-1);
20     end
21 end
22
23 % Display the resulting matrix
24 disp('The generated matrix is:');
25 disp(A);
26

```

The command window displays the output of the script. It prompts the user for 'n' and 'm', both set to 4. It then displays the generated matrix:

```

Enter the number of rows (n): 4
Enter the number of columns (m): 5
The generated matrix is:
1 2 3 4 5
2 4 7 11 16
3 7 14 25 41
4 11 25 50 91
>>

```

5. A person in retirement is depositing \$300,000 in a saving account that pays 5% interest per year. The person plans to withdraw money from the account once a year. He starts by withdrawing \$25,000 after the first year, and in future years he increases the amount he withdraws according to the inflation rate. For example, if the inflation rate is 3%, he withdraws \$25,750 after the second year. Calculate the number of years the money in the account will last assuming a constant yearly inflation rate of 2%. Make a plot that shows the yearly withdrawals and the balance of the account over the years.

The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, CODE, ANALYZE, TEST, and SECTION. Below the menu is a toolbar with icons for New, Open, Save, Compare, Go To, Find, Copilot, Refactor, Debugger, Generate Test, Run Section, Run and Advance, Run to End, Section Break, Run, Step, and Stop. The central workspace contains a script editor window titled 'EXPm' and a command window. The script code is as follows:

```

1 % Retirement account simulation
2 % Parameters
3 initial_balance = 300000;
4 interest_rate = 0.05;
5 withdrawal = 25000;
6 inflation_rate = 0.02;
7
8 % Initialize variables
9 balance = initial_balance;
10 year = 0;
11 balances = [];
12 withdrawals = [];
13
14 % Loop until money runs out
15 while balance > 0
16     year = year + 1;
17
18     % Apply interest
19     balance = balance * (1 + interest_rate);
20
21     % Withdraw money
22     if balance < withdrawal
23         withdrawal = balance; % last withdrawal
24     end
25     balance = balance - withdrawal;
26
27     % Store values for plotting
28     balances(end+1) = balance;
29     withdrawals(end+1) = withdrawal;

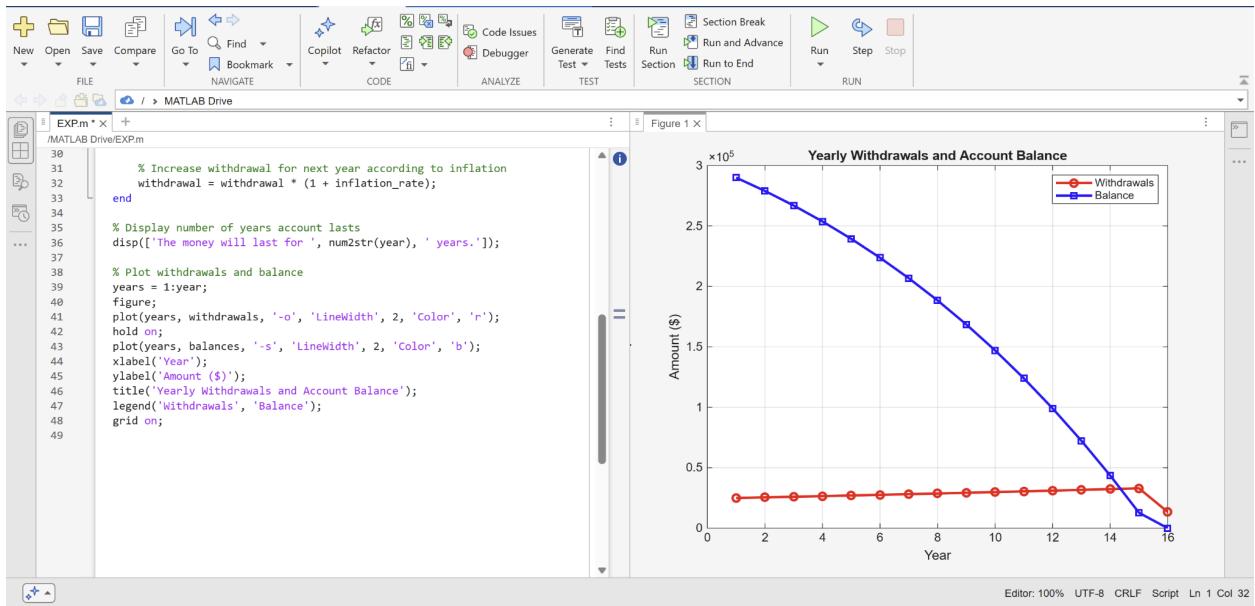
```

The command window displays the output of the script. It informs the user that the money will last for 16 years:

```

The money will last for 16 years.
The money will last for 16 years.
>> Press Ctrl + Shift + P to generate code with Copilot

```



6. Use loops to create a matrix in which the value of each element is the sum of its row number and its column number divided by the square of its column number.

The figure shows the MATLAB IDE interface. On the left is the code editor with a script named EXP.m. The code initializes a matrix A with zeros, fills it using nested loops, and then displays the resulting matrix. On the right is the Command Window. It prompts the user for the number of rows and columns, then displays the generated matrix. The matrix has 3 rows and 4 columns.

```

1 % Initialize matrix
2 A = zeros(n, m);
3
4 % Fill matrix using nested loops
5 for i = 1:n
6     for j = 1:m
7         A(i,j) = (i + j) / (j^2);
8     end
9 end
10
11 % Display the resulting matrix
12 disp('The generated matrix is:');
13 disp(A);

```

Command Window Output:

```

Enter the number of rows: 3
Enter the number of columns: 4
The generated matrix is:
2.0000    0.7500    0.4444    0.3125
3.0000    1.0000    0.5556    0.3750
4.0000    1.2500    0.6667    0.4375
>>

```

7. Write a program in a script file that finds the smallest odd integer that is divisible by 11 and whose square root is greater than 132. Use a loop in the program. The loop should start from 1 and stop when the number is found. The program prints the message “The required number is:” and then prints the number.

The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, NAVIGATE, CODE, ANALYZE, TEST, SECTION, and RUN. The left pane displays a script named EXPm.m with the following code:

```

EXPm.m
1 % Loop until the condition is satisfied
2 while true
3     if mod(n, 2) ~= 0 && mod(n, 11) == 0 && sqrt(n) > 132
4         fprintf('The required number is: %d\n', n);
5         break;
6     end
7     n = n + 1;
8 end

```

The right pane is the Command Window, which shows the output: "The required number is: 17435". A tooltip at the bottom right of the window says: "Press [Ctrl] + [Shift] + [P] to generate code with Copilot". The status bar at the bottom indicates: Editor: 100% UTF-8 CRLF Script Ln 15 Col 1.

8. A vector is given by $x = [-3.5 \ 5 \ -6.2 \ 11.1 \ 0 \ 7 \ -9.5 \ 2 \ 15 \ -1 \ 3 \ 2.5]$. Using conditional statements and loops, write a program that rearranges the elements of x in order from the smallest to the largest. Do not use MATLAB's built-in function sort.

The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, NAVIGATE, CODE, ANALYZE, TEST, SECTION, and RUN. The left pane displays a script named EXPm.m with the following code:

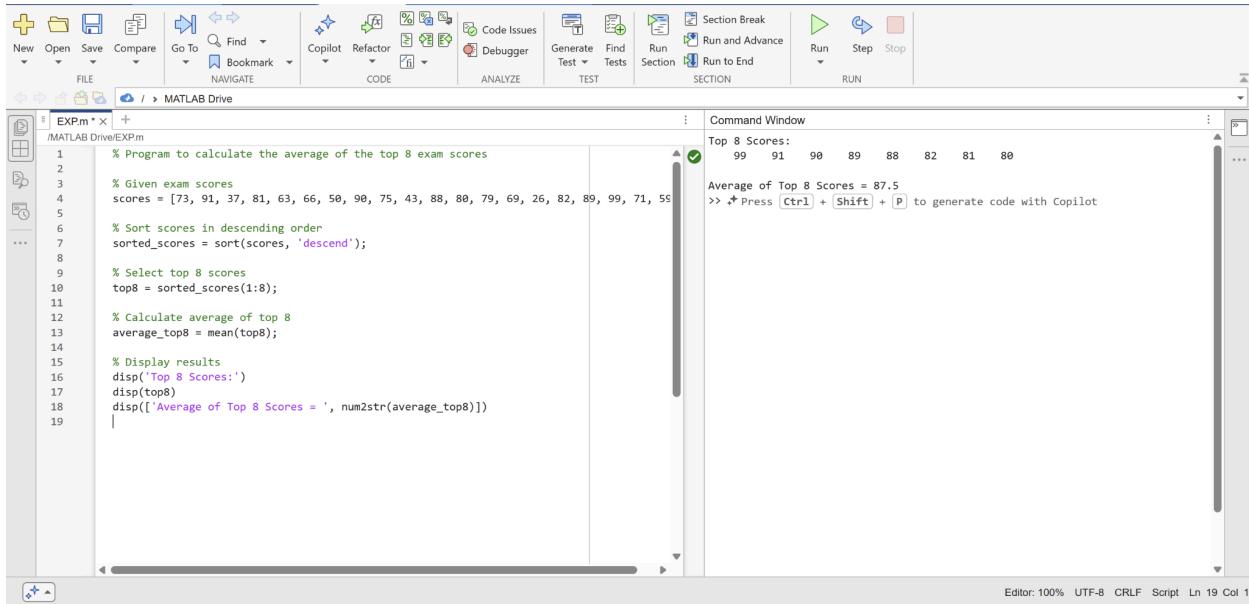
```

EXPm.m
1 % Program to rearrange the elements of a vector in ascending order
2 % Given vector
3 x = [-3.5 5 -6.2 11.1 0 7 -9.5 2 15 -1 3 2.5];
4
5 % Get the length of the vector
6 n = length(x);
7
8 % Sorting using Bubble Sort algorithm
9 for i = 1:n-1
10    for j = 1:n-i
11        if x(j) > x(j+1)
12            % Swap elements if they are in wrong order
13            temp = x(j);
14            x(j) = x(j+1);
15            x(j+1) = temp;
16        end
17    end
18
19    % Display the sorted vector
20    disp('The elements of x arranged from smallest to largest are:')
21    disp(x)
22

```

The right pane is the Command Window, which shows the output: "The elements of x arranged from smallest to largest are: Columns 1 through 8 -9.5000 -6.2000 -3.5000 -1.0000 0 2.0000 2.5000 3.0000 Columns 9 through 12 5.0000 7.0000 11.1000 15.0000". A tooltip at the bottom right of the window says: "Press [Ctrl] + [Shift] + [P] to generate code with Copilot". The status bar at the bottom indicates: Editor: 100% UTF-8 CRLF Script Ln 1 Col 67.

9. The following is a list of 20 exam scores. Write a computer program that calculates the average of the top 8 scores. Exam scores: 73, 91, 37, 81, 63, 66, 50, 90, 75, 43, 88, 80, 79, 69, 26, 82, 89, 99, 71, 59



The screenshot shows the MATLAB IDE interface. The left pane displays the code in a script named EXP.m:

```

1 % Program to calculate the average of the top 8 exam scores
2
3 % Given exam scores
4 scores = [73, 91, 37, 81, 63, 66, 50, 90, 75, 43, 88, 80, 79, 69, 26, 82, 89, 99, 71, 59];
5
6 % Sort scores in descending order
7 sorted_scores = sort(scores, 'descend');
8
9 % Select top 8 scores
10 top8 = sorted_scores(1:8);
11
12 % Calculate average of top 8
13 average_top8 = mean(top8);
14
15 % Display results
16 disp('Top 8 Scores:');
17 disp(top8);
18 disp(['Average of Top 8 Scores = ', num2str(average_top8)]);
19

```

The right pane shows the Command Window output:

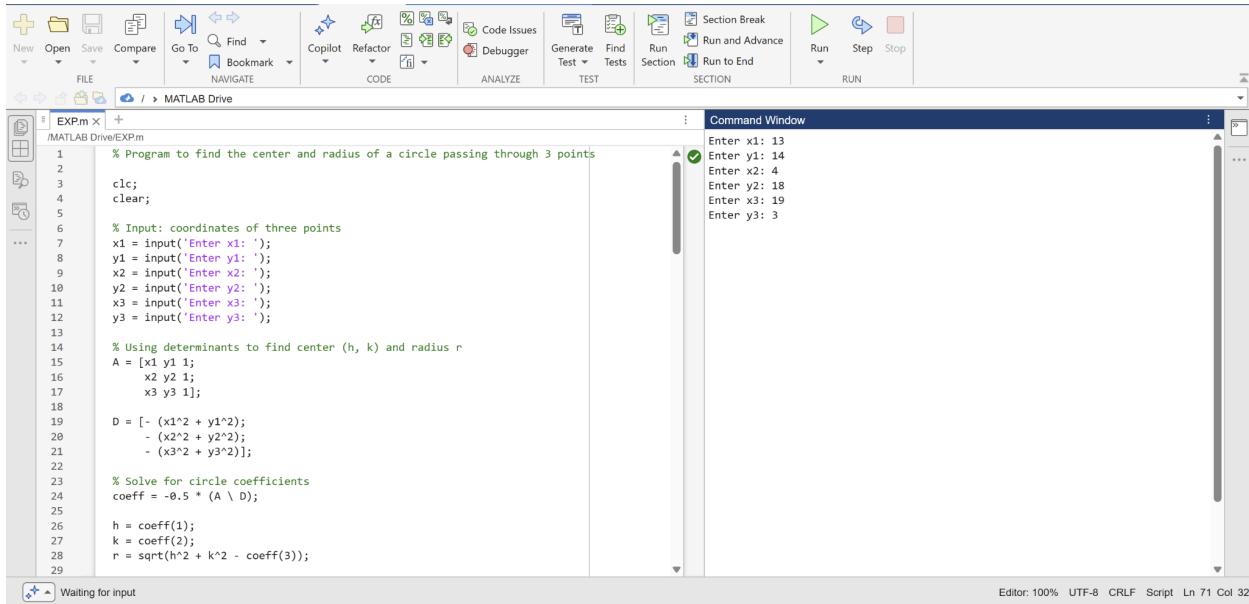
```

Top 8 Scores:
99 91 90 89 88 82 81 80
Average of Top 8 Scores = 87.5
>> * Press Ctrl + Shift + P to generate code with Copilot

```

At the bottom, the status bar indicates: Editor: 100% UTF-8 CRLF Script Ln 19 Col 1

10. Write a program that determines the center and the radius of a circle that passes through three given points. The program asks the user to enter the coordinates of the points one at a time. The program displays the coordinate of the center and the radius, and makes a plot of the circle and the three points displayed on the plot with asterisk markers. Execute the program to find the circle that passes through the points (13, 15), (4, 18), and (19, 3).



The screenshot shows the MATLAB IDE interface. The left pane displays the code in a script named EXP.m:

```

1 % Program to find the center and radius of a circle passing through 3 points
2
3 clc;
4 clear;
5
6 % Input: coordinates of three points
7 x1 = input('Enter x1: ');
8 y1 = input('Enter y1: ');
9 x2 = input('Enter x2: ');
10 y2 = input('Enter y2: ');
11 x3 = input('Enter x3: ');
12 y3 = input('Enter y3: ');
13
14 % Using determinants to find center (h, k) and radius r
15 A = [x1 y1 1;
16     x2 y2 1;
17     x3 y3 1];
18
19 D = [- (x1^2 + y1^2);
20       - (x2^2 + y2^2);
21       - (x3^2 + y3^2)];
22
23 % Solve for circle coefficients
24 coeff = -0.5 * (A \ D);
25
26 h = coeff(1);
27 k = coeff(2);
28 r = sqrt(h^2 + k^2 - coeff(3));
29

```

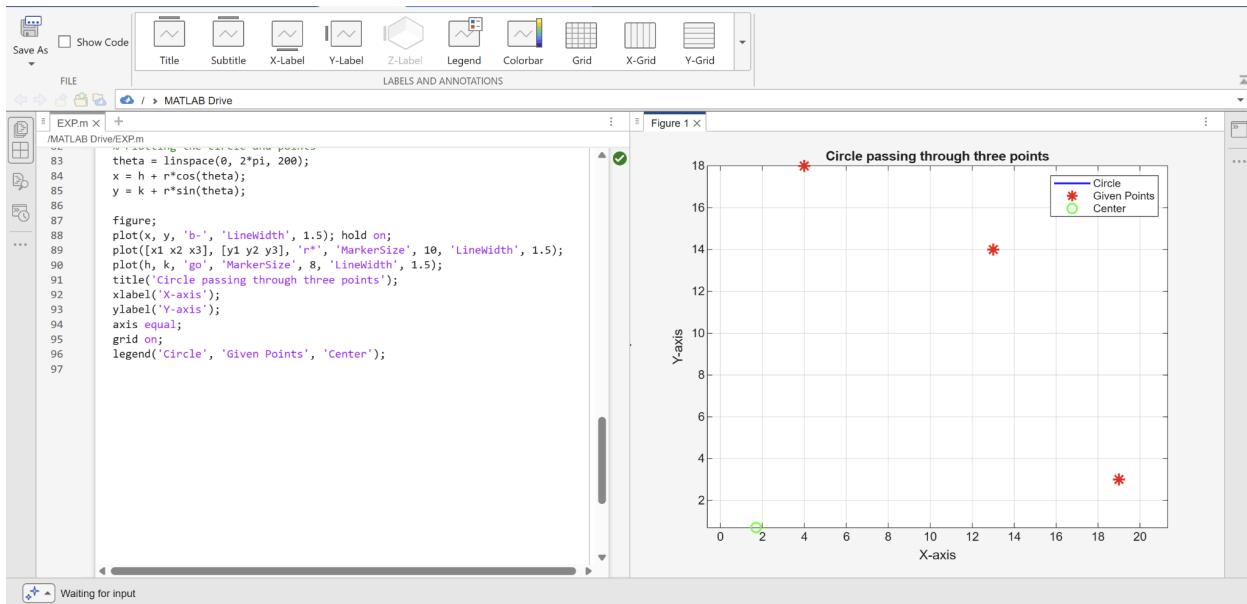
The right pane shows the Command Window output, indicating it is waiting for input:

```

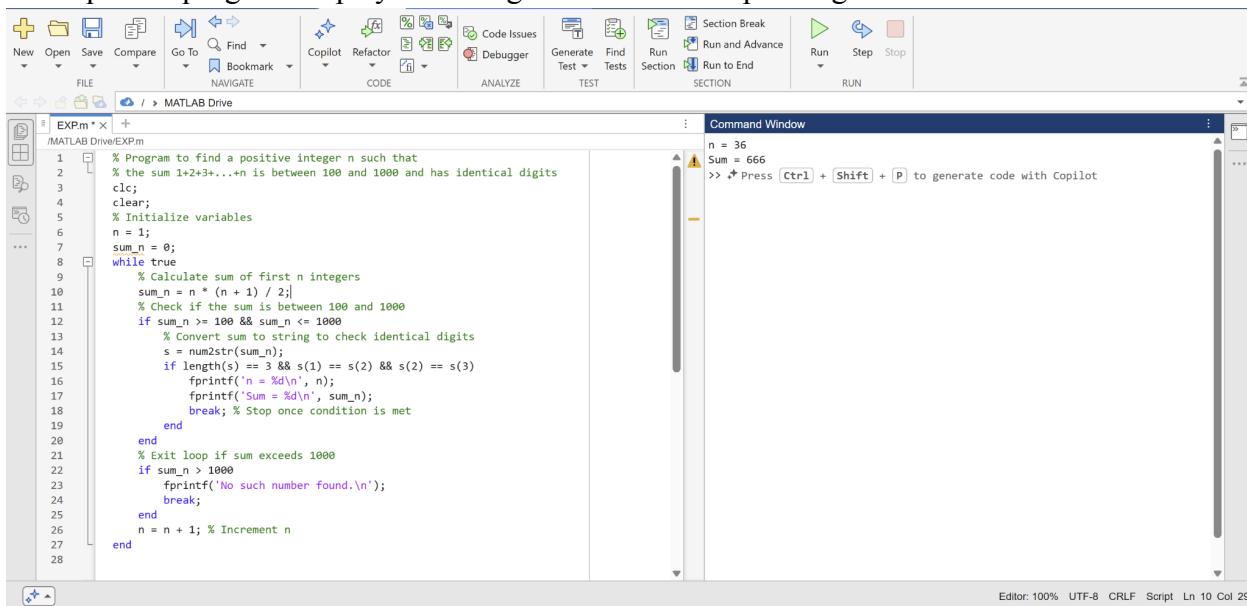
Enter x1: 13
Enter y1: 14
Enter x2: 4
Enter y2: 18
Enter x3: 19
Enter y3: 3

```

At the bottom, the status bar indicates: Editor: 100% UTF-8 CRLF Script Ln 71 Col 32



11. Write a MATLAB program in a script file that finds a positive integer n such that the sum of all the integers $1+2+3+\dots+n$ is a number between 100 and 1000 whose three digits are identical. As output the program displays the integer n and the corresponding sum.



12. A twin primes is a pair of prime numbers such that the difference between them is 2 (for example, 17 and 19). Write a MATLAB program that finds all the twin primes between 10 and 500. The program displays the results in a two-column matrix in which each row is a twin prime.

The screenshot shows the MATLAB IDE interface. The top menu bar includes FILE, CODE, ANALYZE, TEST, SECTION, and RUN. The left pane shows a file browser with 'EXP.m' selected. The right pane is the Command Window displaying the results of the MATLAB script. The script code is as follows:

```
% Program to find all twin primes between 10 and 500
clc;
clear;

% Define range
lower_limit = 10;
upper_limit = 500;

% Initialize an empty array to store twin primes
twin_primes = [];

% Loop through numbers in range
for n = lower_limit:upper_limit
    % Check if both n and n+2 are prime
    if isprime(n) && isprime(n + 2)
        twin_primes = [twin_primes; n, n + 2];
    end
end

% Display the results
disp('Twin Primes between 10 and 500:');
disp(twin_primes);
```

The Command Window output lists pairs of twin primes between 10 and 500:

Prime 1	Prime 2
11	13
17	19
29	31
41	43
59	61
71	73
101	103
167	169
137	139
149	151
179	181
191	193
197	199
227	229
239	241
269	271
281	283
311	313
347	349
419	421
431	433
461	463

Editor: 100% UTF-8 CRLF Script Ln 23 Col 1