VIDYAVARDHAKA COLLEGE OF ENGINEERING
DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

DEPARTMENT OF MATHEMATICS

Academic Year 2024 - 2025

Mathematics-IV for IT Stream

Subject Code: BITMA401

ABA

Submitted By:

M R Padmanabha

4VV23IS107

IV IS-B

VVCE Mysuru.

Submitted To:

Dr. Divya Rashmi,

Department of Mathematics,

VVCE Mysuru.

QUESTION 1:

Given the data:

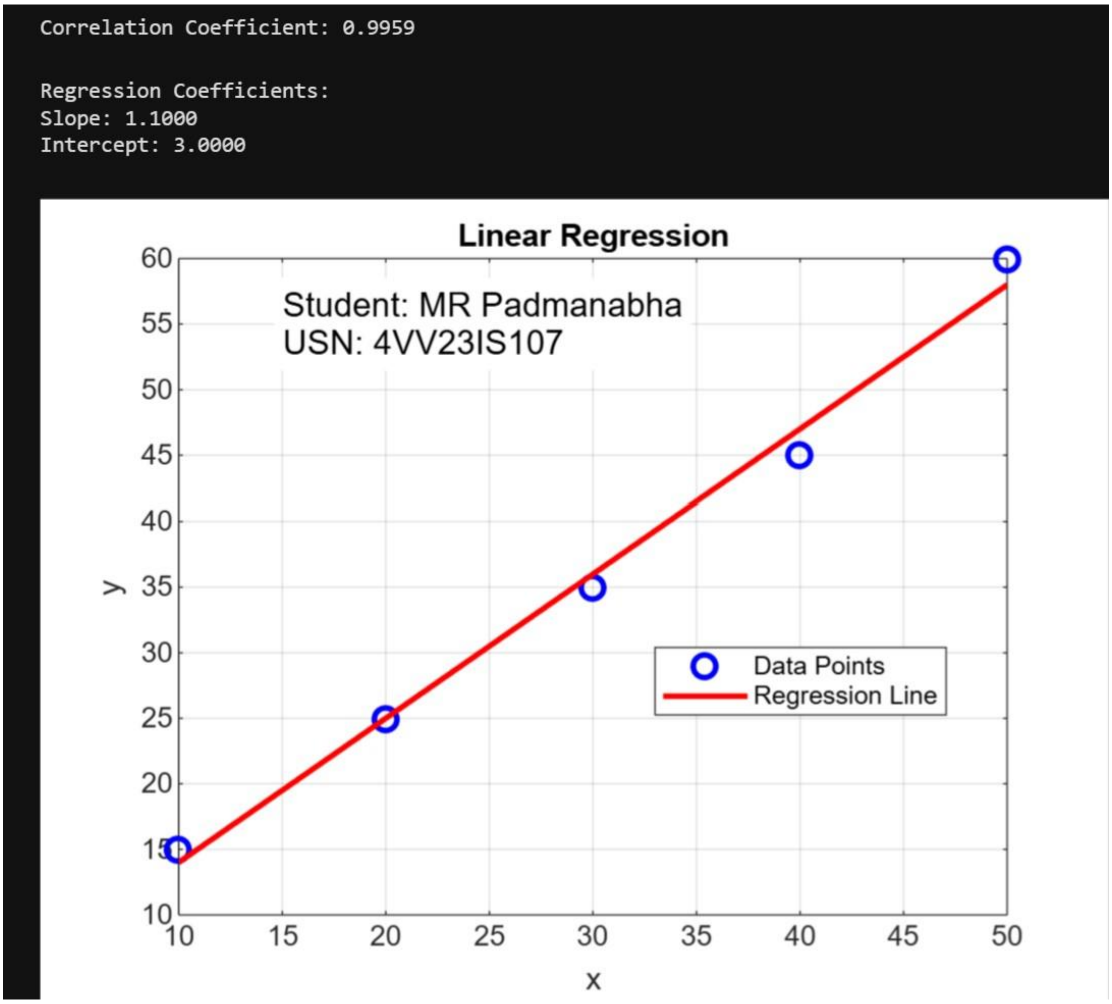
x	10	20	30	40	50
y	15	25	35	45	60

Write a MATLAB script to:

i. Calculate the correlation coefficient.
ii. Fit a linear regression line.
iii. Compute and display the regression coefficients.
iv. Plot the data and regression line.

```
% Define data points
x = [10 20 30 40 50];
y = [15 25 35 45 60];

% Define student details
student_name = 'MR Padmanabha'; % Change this as needed
usn = '4VV23IS107'; % Change this as needed

% Calculate the correlation coefficient
r = corrcoef(x, y);
fprintf('Correlation Coefficient: %.4f\n', r(1,2));

% Fit a linear regression line
p = polyfit(x, y, 1);
m = p(1); % Slope
c = p(2); % Intercept

% Display regression coefficients
fprintf('Regression Coefficients:\nSlope: %.4f\nIntercept: %.4f\n', m, c);

% Generate fitted values
y_fit = polyval(p, x);

% Plot data points and regression line
figure;
plot(x, y, 'bo', 'MarkerSize', 8, 'LineWidth', 2); % Plot original data
hold on;
plot(x, y_fit, 'r-', 'LineWidth', 2); % Plot regression line
xlabel('x');
ylabel('y');
title('Linear Regression', 'Units', 'normalized', 'Position', [0.5, 1.05], 'FontSize',
14);

% Adjust placement of student details to a better position
annotation_text = sprintf('Student: %s\nUSN: %s', student_name, usn);
text(min(x) + 5, max(y) - 5, annotation_text, 'FontSize', 12, 'Color', 'black',
'BackgroundColor', 'white');
```

```
legend('Data Points', 'Regression Line', 'Location', 'best');
grid on;
hold off;
```



QUESTION 2:

You are given the following data points:

x	1	2	3	4	5
y	2.2	2.8	3.6	4.5	5.1

Write a MATLAB script to perform the following:

i. Fit a **linear model** $y = a + bx$ to the data using the **least squares method** (do not use polyfit).

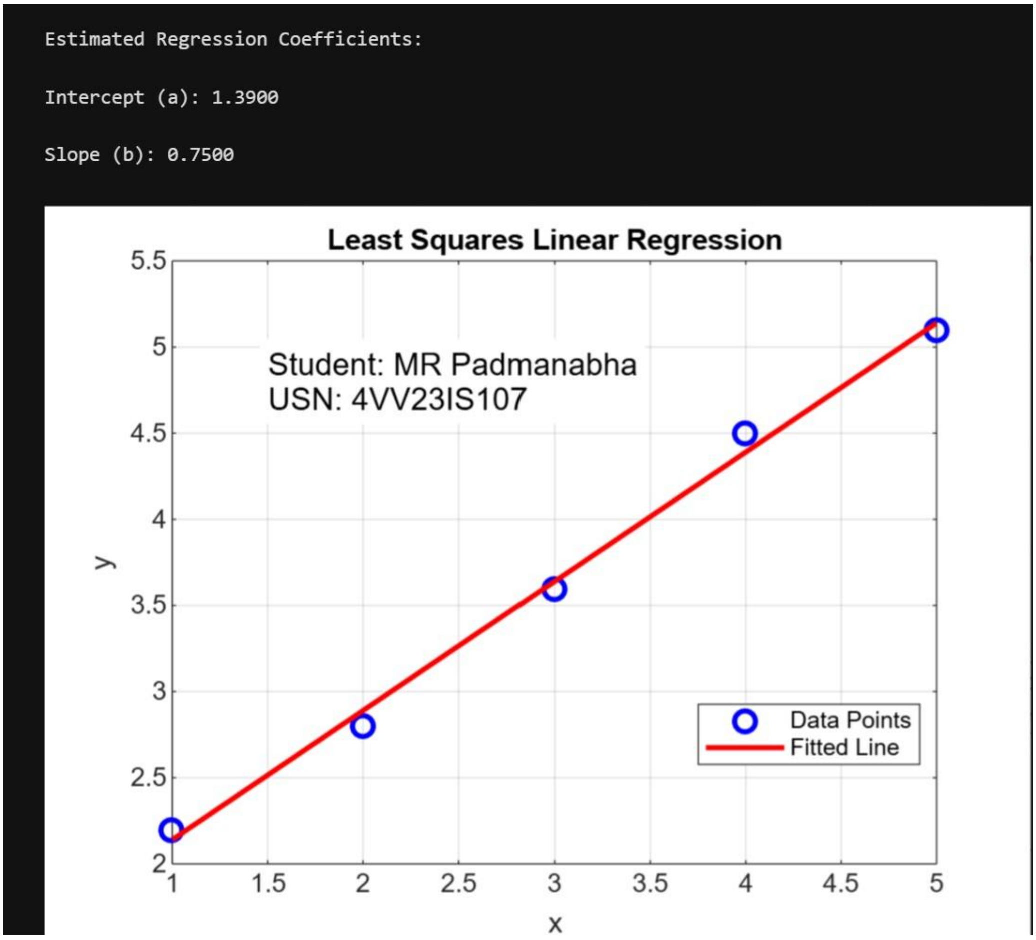
ii. Plot the original data points and the fitted line on the same graph.

iii. Display the estimated values of a (intercept) and b(slope).

```
% Define data points
x = [1 2 3 4 5];
y = [2.2 2.8 3.6 4.5 5.1];
```

```
% Define student details
student_name = 'MR Padmanabha'; % Change this as needed
usn = '4VV23IS107'; % Change this as needed
```

```matlab
% Number of data points
n = length(x);

% Compute summations needed for least squares method
sum_x = sum(x);
sum_y = sum(y);
sum_xy = sum(x .* y);
sum_x2 = sum(x .^ 2);

% Calculate slope (b) and intercept (a)
b = (n * sum_xy - sum_x * sum_y) / (n * sum_x2 - sum_x^2);
a = (sum_y - b * sum_x) / n;

% Display results
fprintf('Estimated Regression Coefficients:\n');
fprintf('Intercept (a): %.4f\n', a);
fprintf('Slope (b): %.4f\n', b);

% Compute fitted values
y_fit = a + b * x;

% Plot original data points and regression line
figure;
plot(x, y, 'bo', 'MarkerSize', 8, 'LineWidth', 2); % Data points
hold on;
plot(x, y_fit, 'r-', 'LineWidth', 2); % Fitted regression line
xlabel('x');
ylabel('y');
title('Least Squares Linear Regression');

% Adjust placement of student details for better visibility
annotation_text = sprintf('Student: %s\nUSN: %s', student_name, usn);
text(min(x) + 0.5, max(y) - 0.3, annotation_text, 'FontSize', 12, 'Color', 'black',
'BackgroundColor', 'white');

legend('Data Points', 'Fitted Line', 'Location', 'best');
grid on;
hold off;
```



QUESTION 3:

You are given the following data points:

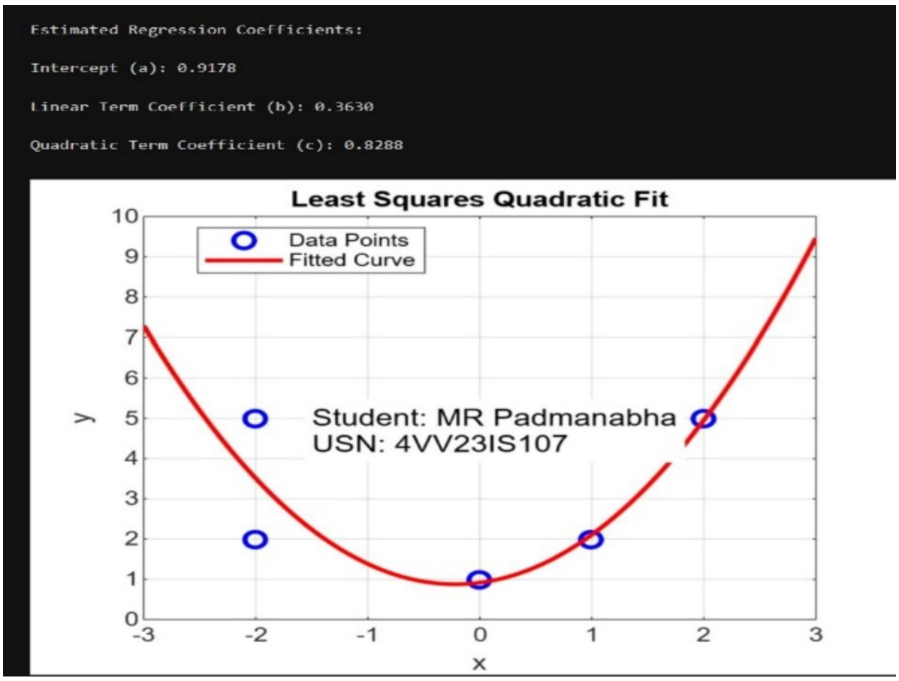
x	-2	-2	0	1	2
y	5	2	1	2	5

Write a MATLAB script to:

i. Fit a **second-degree polynomial** (parabola) of the form $y = a + bx + cx^2$ to the given data using the **least squares method** (do **not** use polyfit).
ii. Plot the original data points and the fitted curve on the same graph.
iii. Display the estimated values of a, b, and c.

```
% Define data points
  x = [-2 -2 0 1 2];
  y = [5 2 1 2 5];


  % Define student details
  student_name = 'MR Padmanabha'; % Change this as needed
  usn = '4VV23IS107'; % Change this as needed
% Number of data points
  n = length(x);

  % Construct matrix A and vector B for solving the normal equations
  A = [n sum(x) sum(x.^2); sum(x) sum(x.^2) sum(x.^3); sum(x.^2) sum(x.^3) sum(x.^4)];
  B = [sum(y); sum(x.*y); sum(x.^2.*y)];
```

```
% Solve for coefficients a, b, and c
coeffs = A \ B;
a = coeffs(1);
b = coeffs(2);
c = coeffs(3);

% Display results
fprintf('Estimated Regression Coefficients:\n');
fprintf('Intercept (a): %.4f\n', a);
fprintf('Linear Term Coefficient (b): %.4f\n', b);
fprintf('Quadratic Term Coefficient (c): %.4f\n', c);

% Generate fitted values for smooth plotting
x_fit = linspace(min(x)-1, max(x)+1, 100);
y_fit = a + b*x_fit + c*x_fit.^2;

% Plot original data points and regression curve
figure;
plot(x, y, 'bo', 'MarkerSize', 8, 'LineWidth', 2); % Data points
hold on;
plot(x_fit, y_fit, 'r-', 'LineWidth', 2); % Fitted curve
xlabel('x');
ylabel('y');
title('Least Squares Quadratic Fit');

% Adjust placement of student details for better visibility
annotation_text = sprintf('Student: %s\nUSN: %s', student_name, usn);
text(min(x) + 0.5, max(y) - 0.3, annotation_text, 'FontSize', 12, 'Color', 'black',
'BackgroundColor', 'white');

legend('Data Points', 'Fitted Curve', 'Location', 'best');
grid on;
hold off;
```



QUESTION 4:

Given a discrete random variable X with values and probabilities:

X	0	1	2	3
$P(X)$	0.1	0.2	0.4	0.3

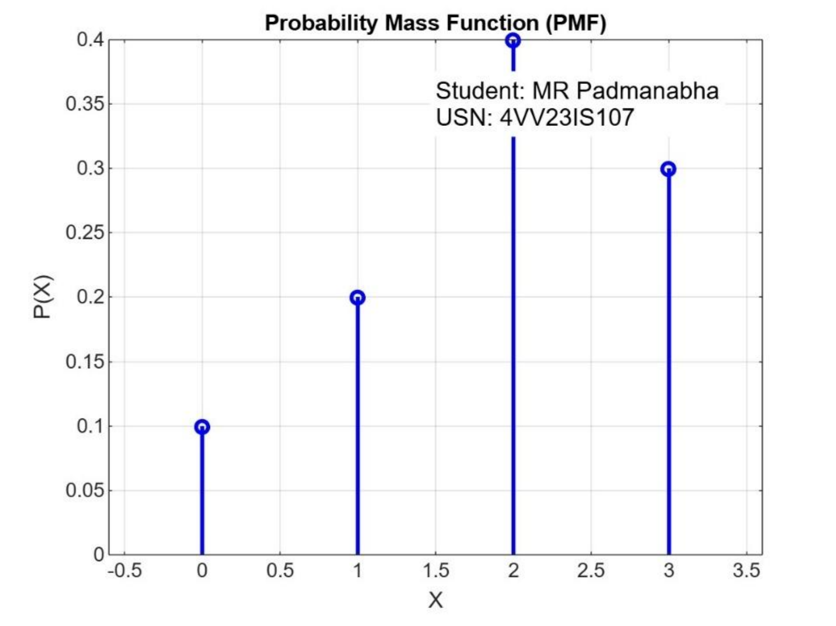
i. Write a MATLAB script to verify that p(x) is a valid probability mass function (PMF).
ii. Compute and display the cumulative distribution function (CDF).
iii. Plot both the PMF and CDF on separate graphs.
iv. Calculate the expected value and variance of X.

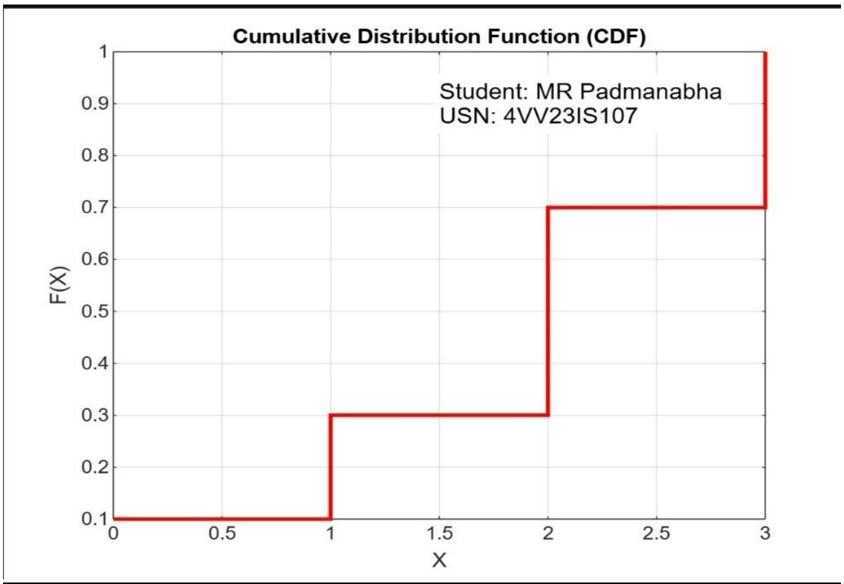
```
% Define discrete random variable X and its corresponding probabilities
X = [0 1 2 3];
P_X = [0.1 0.2 0.4 0.3];

% Define student details
student_name = 'MR Padmanabha'; % Change this as needed
usn = '4VV23IS107'; % Change this as needed

% Verify that P(X) is a valid PMF (sum must be 1)
pmf_sum = sum(P_X);
if abs(pmf_sum - 1) < 1e-6
    fprintf('P(X) is a valid probability mass function.\n');
else
    fprintf('P(X) is NOT a valid probability mass function.\n');
end

% Compute cumulative distribution function (CDF)
CDF = cumsum(P_X);

% Display the CDF values
fprintf('\nCumulative Distribution Function (CDF):\n');
for i = 1:length(X)
    fprintf('F(%d) = %.2f\n', X(i), CDF(i));
end

% Compute expected value (E[X]) and variance (Var[X])
E_X = sum(X .* P_X);
E_X2 = sum((X .^ 2) .* P_X);
Var_X = E_X2 - E_X^2;

% Display expected value and variance
fprintf('\nExpected Value (E[X]): %.4f\n', E_X);
fprintf('Variance (Var[X]): %.4f\n', Var_X);

% Plot PMF
figure;
```

```
stem(X, P_X, 'bo', 'LineWidth', 2);
xlabel('X');
ylabel('P(X)');
title('Probability Mass Function (PMF)');
% Add student details to PMF plot
annotation_text = sprintf('Student: %s\nUSN: %s', student_name, usn);
text(1.5, max(P_X)-0.05, annotation_text, 'FontSize', 12, 'Color', 'black',
'BackgroundColor', 'white');
 grid on;
```

```
% Plot CDF
figure;
stairs(X, CDF, 'r-', 'LineWidth', 2);
xlabel('X');
ylabel('F(X)');
title('Cumulative Distribution Function (CDF)');
% Add student details to CDF plot
text(1.5, max(CDF)-0.1, annotation_text, 'FontSize', 12, 'Color', 'black',
'BackgroundColor', 'white');
 grid on;
```

```
P(X) is a valid probability mass function.

Cumulative Distribution Function (CDF):

F(0) = 0.10
F(1) = 0.30
F(2) = 0.70
F(3) = 1.00

Expected Value (E[X]): 1.9000

Variance (Var[X]): 0.8900
```





QUESTION 5:

Given the joint PMF of two discrete random variables X and Y:

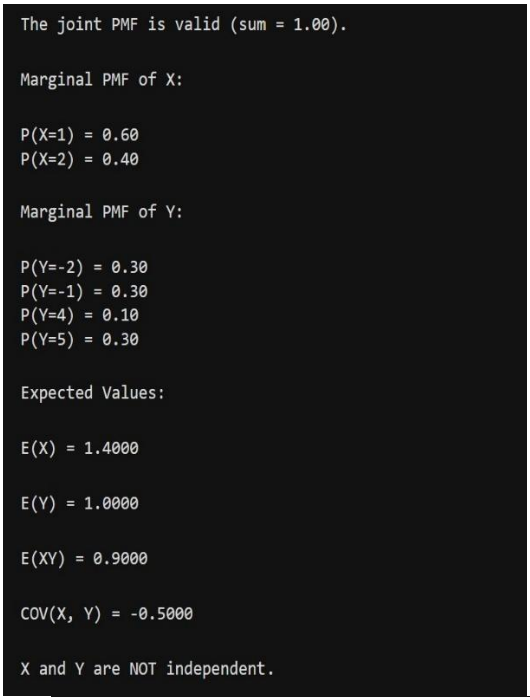
$X \setminus Y$	-2	-1	4	5
1	0.1	0.2	0.0	0.3
2	0.2	0.1	0.1	0.0

i. Verify that the joint distribution is valid.
ii. Compute and display the marginal PMFs of X and Y.
iii. Compute $E(X), E(Y), and\ E(XY), COV(X,Y)$, σ_X and σ_Y
iv. Determine whether X and Y are independent.

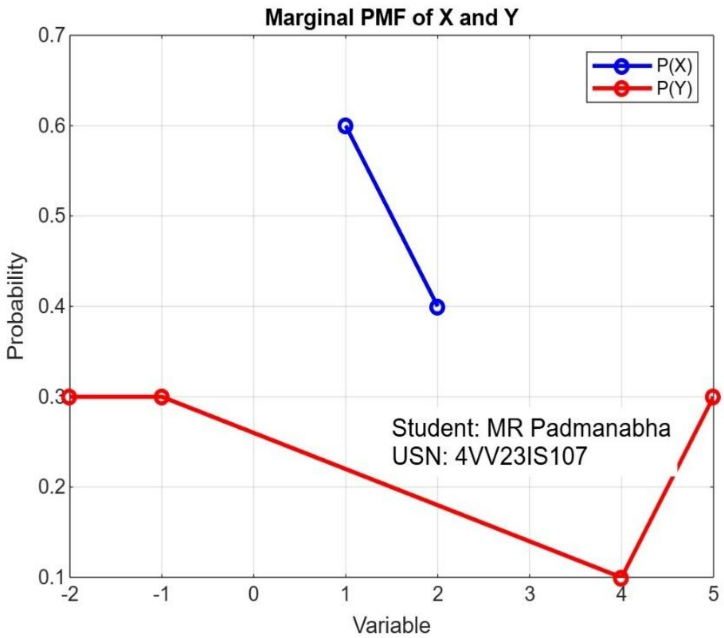
```
% Define discrete random variables X and Y
X = [1 2];
Y = [-2 -1 4 5];

% Define student details
student_name = 'MR Padmanabha'; % Change this as needed
usn = '4VV23IS107'; % Change this as needed

% Define the joint probability mass function (PMF)
P_XY = [0.1 0.2 0.0 0.3;  % Row corresponding to X = 1
        0.2 0.1 0.1 0.0];  % Row corresponding to X = 2

% Verify if the joint PMF is valid (sum must be 1)
pmf_sum = sum(P_XY, 'all');
if abs(pmf_sum - 1) < 1e-6
    fprintf('The joint PMF is valid (sum = %.2f).\n', pmf_sum);
else
    fprintf('The joint PMF is NOT valid (sum = %.2f).\n', pmf_sum);
end

% Compute marginal PMFs
P_X = sum(P_XY, 2); % Summing over Y to get P(X)
P_Y = sum(P_XY, 1); % Summing over X to get P(Y)

% Display marginal PMFs
fprintf('\nMarginal PMF of X:\n');
for i = 1:length(X)
    fprintf('P(X=%d) = %.2f\n', X(i), P_X(i));
end

fprintf('\nMarginal PMF of Y:\n');
for j = 1:length(Y)
    fprintf('P(Y=%d) = %.2f\n', Y(j), P_Y(j));
end
% Compute expectations
E_X = sum(X .* P_X');
```

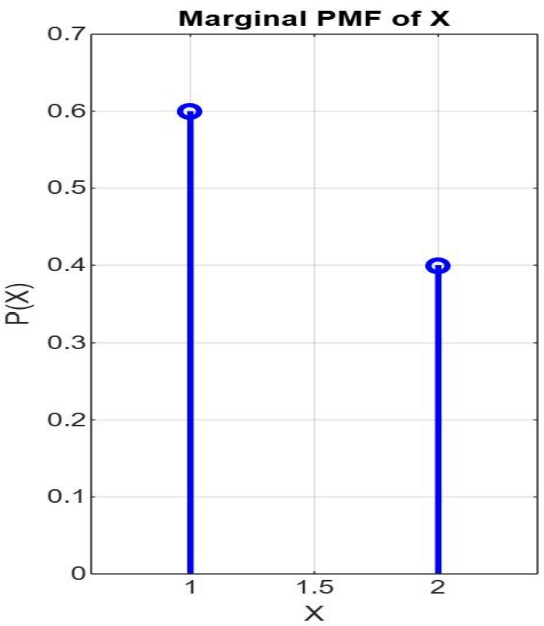
```
E_Y = sum(Y .* P_Y);

% Compute E(XY)
E_XY = sum((X' * Y) .* P_XY, 'all');

% Compute covariance: COV(X,Y) = E(XY) - E(X)E(Y)
COV_XY = E_XY - (E_X * E_Y);

% Display results
fprintf('\nExpected Values:\n');
fprintf('E(X) = %.4f\n', E_X);
fprintf('E(Y) = %.4f\n', E_Y);
fprintf('E(XY) = %.4f\n', E_XY);
fprintf('COV(X, Y) = %.4f\n', COV_XY);

% Check independence: If P(X,Y) = P(X)P(Y) for all values, they are independent
independent = true;
for i = 1:length(X)
    for j = 1:length(Y)
        if abs(P_XY(i,j) - P_X(i) * P_Y(j)) > 1e-6
            independent = false;
            break;
        end
    end
end

if independent
    fprintf('\nX and Y are independent.\n');
else
    fprintf('\nX and Y are NOT independent.\n');
end

% Plot Marginal PMFs
figure;
subplot(1, 2, 1);
stem(X, P_X, 'bo', 'LineWidth', 2);
xlabel('X');
ylabel('P(X)');
title('Marginal PMF of X');
grid on;

subplot(1, 2, 2);
stem(Y, P_Y, 'ro', 'LineWidth', 2);
xlabel('Y');
ylabel('P(Y)');
title('Marginal PMF of Y');
grid on;

% Add student details to the plot
annotation_text = sprintf('Student: %s\nUSN: %s', student_name, usn);
```

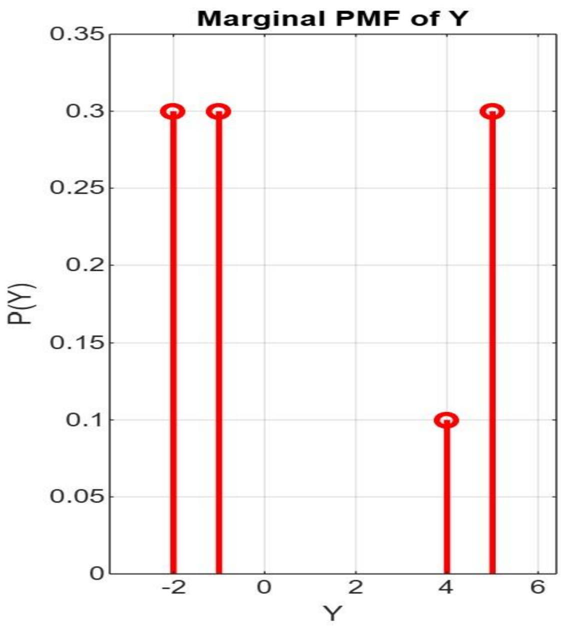
```
text(1.5, max(P_X) - 0.05, annotation_text, 'FontSize', 12, 'Color', 'black',
'BackgroundColor', 'white');

figure;
plot(X, P_X, 'bo-', 'LineWidth', 2);
hold on;
plot(Y, P_Y, 'ro-', 'LineWidth', 2);
xlabel('Variable');
ylabel('Probability');
title('Marginal PMF of X and Y');
legend('P(X)', 'P(Y)');
grid on;
% Add student details to the second plot
text(1.5, max(P_Y) - 0.05, annotation_text, 'FontSize', 12, 'Color', 'black',
'BackgroundColor', 'white'); hold off;
```



```
The joint PMF is valid (sum = 1.00).

Marginal PMF of X:

P(X=1) = 0.60
P(X=2) = 0.40

Marginal PMF of Y:

P(Y=-2) = 0.30
P(Y=-1) = 0.30
P(Y=4) = 0.10
P(Y=5) = 0.30

Expected Values:

E(X) = 1.4000

E(Y) = 1.0000

E(XY) = 0.9000

COV(X, Y) = -0.5000

X and Y are NOT independent.
```







QUESTION 6:

You are given the matrix: $P = \begin{pmatrix} 0.5 & 0.3 & 0.2 \\ 0.2 & 0.6 & 0.2 \\ 0.3 & 0.3 & 0.4 \end{pmatrix}$

i. Write a MATLAB function or script to check whether the matrix is a **stochastic matrix** (i.e., each row sums to 1).

ii. Check if the matrix is **regular**, i.e., find the smallest $k \leq 10$ such that all entries of P^k are positive.

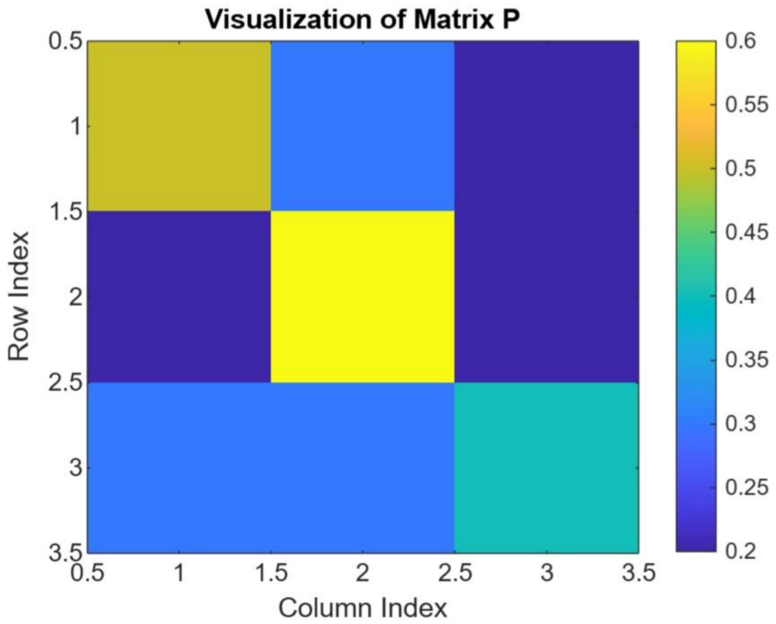
iii. Find the **unique fixed probability vector** π such that πP=π, with $\sum \pi_i = 1$ Use either matrix algebra or the null function in MATLAB.

iv. Display the results with appropriate comments.

```
% Define matrix P
P = [0.5 0.3 0.2;
     0.2 0.6 0.2;
     0.3 0.3 0.4];

% Task 1: Check if P is a stochastic matrix (each row sums to 1)
row_sums = sum(P, 2);
is_stochastic = all(abs(row_sums - 1) < 1e-6); % Allow small tolerance for numerical
precision

if is_stochastic
    fprintf('The matrix P is a stochastic matrix.\n');
else
    fprintf('The matrix P is NOT a stochastic matrix.\n');
end

% Task 2: Check for regularity (find smallest k such that all entries of P^k are
positive)
regular_k = -1; % Default value if no k ≤ 10 satisfies condition
for k = 1:10
    P_k = P^k;
    if all(P_k(:) > 0) % Check if all elements are positive
        regular_k = k;
        break;
    end
end

if regular_k ~= -1
    fprintf('The matrix P is regular for k = %d.\n', regular_k);
else
    fprintf('The matrix P is NOT regular for any k ≤ 10.\n');
end

% Task 3: Find the fixed probability vector π such that πP = π and sum(π) = 1
% Solve (P' - I) * π = 0 using the null function
A = (P' - eye(size(P))); % Transpose to solve for left eigenvector
pi_vector = null(A, 'r'); % Find null space
pi_vector = pi_vector / sum(pi_vector); % Normalize so sum(pi) = 1
```

```
% Display results
fprintf('The unique fixed probability vector π is:\n');
disp(pi_vector);

% Task 4: Visualize matrix P using a heatmap for clarity
figure;
imagesc(P); % Display matrix as heatmap
colorbar; % Add color scale
xlabel('Column Index');
ylabel('Row Index');
title('Visualization of Matrix P');

hold off;
```

```
The matrix P is a stochastic matrix.

The matrix P is regular for k = 1.

The unique fixed probability vector π is:

    0.3214
    0.4286
    0.2500
```



QUESTION 7:

A company claims that the average lifetime of its LED bulbs is **1000 hours**. To verify this, a random sample of 15 bulbs was tested, and the lifetimes (in hours) were recorded as

980	1020	995	1005	990	1008	1012	985	998	1002	1015	987	992	1000	1003

i. Use a **one-sample *t*-test** to test the null hypothesis $H_0 : \mu = 1000$ against the alternative $H_1 : \mu \neq 1000$, at a **5% significance level**.

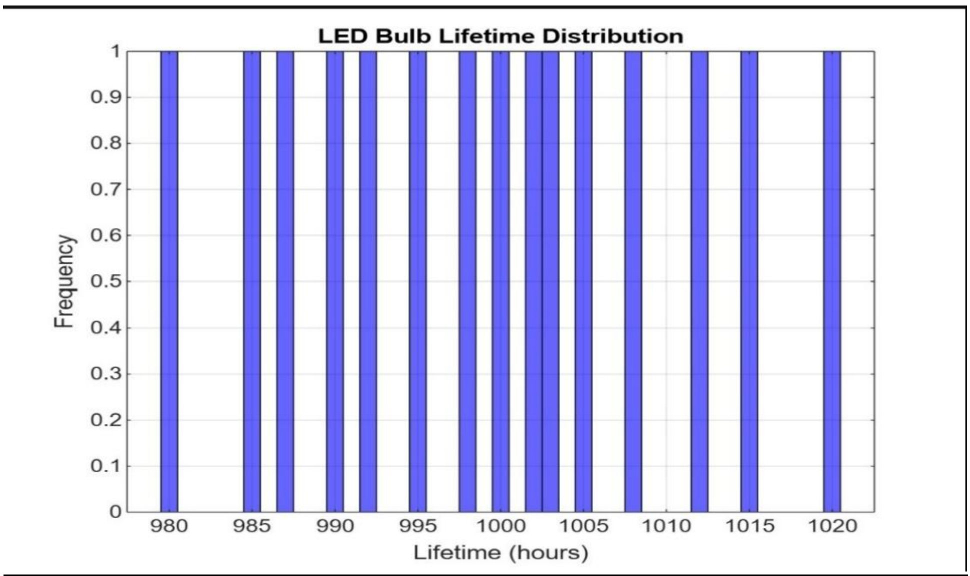
ii. Write a MATLAB script to:
- Perform the *t*-test using MATLAB's ttest function.
- Display the *t*-statistic, p-value, and test decision.

iii. Interpret the result: Should the null hypothesis be rejected or not?

```
% Given sample data: LED bulb lifetimes (hours)
lifetimes = [980 1020 995 1005 990 1008 1012 985 998 1002 1015 987 992 1000 1003];

% Null hypothesis: H0: mu = 1000
mu0 = 1000; % Claimed mean
```

```matlab
% Perform one-sample t-test at 5% significance level
[H, p, ci, stats] = ttest(lifetimes, mu0, 'Alpha', 0.05);
% Display results
fprintf('Results of One-Sample t-test:\n');
fprintf('T-statistic = %.4f\n', stats.tstat);
fprintf('p-value = %.4f\n', p);
fprintf('Confidence Interval: [%.4f, %.4f]\n', ci(1), ci(2));
% Interpret test decision
if H == 1
    fprintf('Reject the null hypothesis (H0). The mean lifetime differs significantly from 1000 hours.\n');
else
    fprintf('Fail to reject the null hypothesis (H0). There is insufficient evidence to conclude the mean lifetime differs.\n');
end

% Visualize sample data with a histogram
figure;
histogram(lifetimes, 'FaceColor', 'blue'); % Histogram of lifetimes
xlabel('Lifetime (hours)');
ylabel('Frequency');
title('LED Bulb Lifetime Distribution');
grid on;

% Define student details
student_name = 'MR Padmanabha'; % Change this as needed
usn = '4VV23IS107'; % Change this as needed

% Adjust placement of student details for correct visibility
annotation_text = sprintf('Student: %s\nUSN: %s', student_name, usn);
text(mean(lifetimes), max(histcounts(lifetimes)) + 1, annotation_text, 'FontSize', 12, 'Color', 'black', 'BackgroundColor', 'white', 'HorizontalAlignment', 'left');
hold off;
```

```
Results of One-Sample t-test:

T-statistic = -0.1795

p-value = 0.8601

Confidence Interval: [993.0943, 1005.8390]

Fail to reject the null hypothesis (H0). There is insufficient evidence to conclude the mean lifetime differs.
```



QUESTION 8:

The following data shows the number of calls received at a call centre per minute over 100 minutes:

$\boldsymbol{No. of Calls}$ (x)	0	1	2	3	4	5
$Frequency$ (f)	5	15	30	25	15	10

i. Write a MATLAB script to:
   a. Store the observed data (values and frequencies).
   b. Compute the **sample mean** of the data (this will be used as the Poisson parameter λ).

ii. Use the Poisson distribution with estimated λ to compute the **expected frequencies** using poisspdf.

iii. Display a **bar graph** comparing the **observed** and **expected frequencies**.

iv. Comment on the goodness of fit visually.

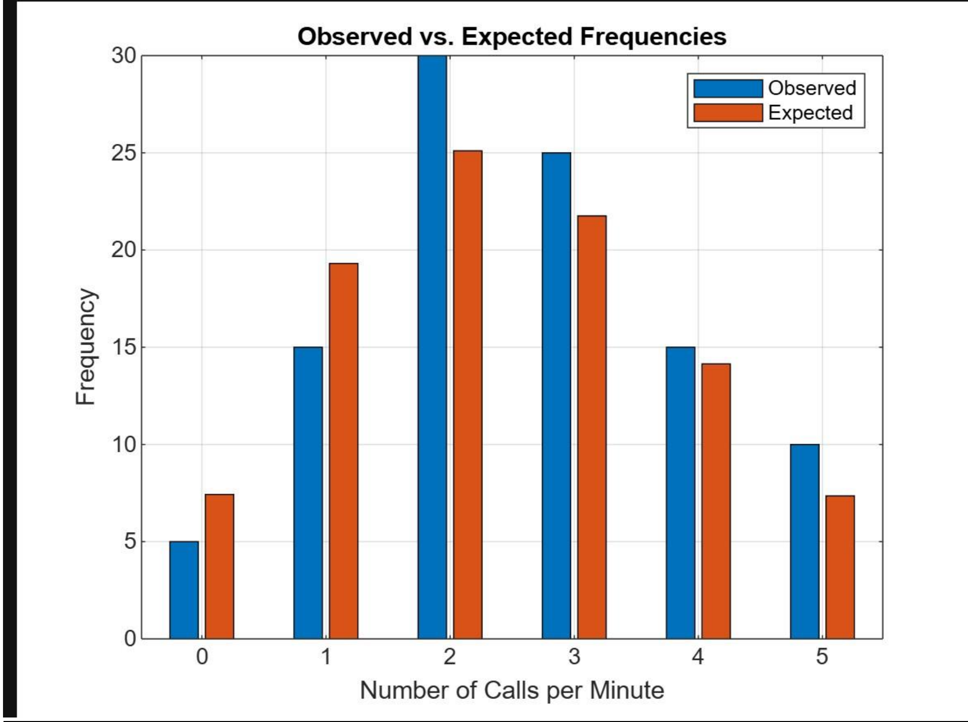
v. *Perform a chi-square goodness-of-fit test.*

```
 % Given observed data: Number of calls per minute and their frequencies
x = [0 1 2 3 4 5]; % Number of calls
f = [5 15 30 25 15 10]; % Frequency of occurrences
n = sum(f); % Total number of observations

% Compute the sample mean (Poisson parameter λ)
lambda = sum(x .* f) / n;
fprintf('Estimated Poisson parameter (λ) = %.4f\n', lambda);

% Compute expected frequencies using Poisson distribution
expected_freq = n * poisspdf(x, lambda);

% Display bar graph comparing observed vs. expected frequencies
figure;
bar(x, [f; expected_freq]', 'grouped');
xlabel('Number of Calls per Minute');
ylabel('Frequency');
title('Observed vs. Expected Frequencies');
legend('Observed', 'Expected');
grid on;

% Perform chi-square goodness-of-fit test
chi2_stat = sum((f - expected_freq).^2 ./ expected_freq);
df = length(x) - 1; % Degrees of freedom
p_value = 1 - chi2cdf(chi2_stat, df);

fprintf('Chi-square test statistic = %.4f\n', chi2_stat);
fprintf('Degrees of freedom = %d\n', df);
fprintf('p-value = %.4f\n', p_value);

% Interpretation of chi-square test
alpha = 0.05; % Significance level
if p_value < alpha
    fprintf('Reject the null hypothesis: The Poisson model does not fit the data
well.\n');
else
```

```
    fprintf('Fail to reject the null hypothesis: The Poisson model fits the data reasonably well.\n');
end
```

```
Chi-square test statistic = 4.1978

Degrees of freedom = 5

p-value = 0.5213

Fail to reject the null hypothesis: The Poisson model fits the data reasonably well.
```

```
Estimated Poisson parameter (λ) = 2.6000
```



QUESTION 9:

Consider the following linear programming problem:

Maximize the objective function: $Z = 3x + 2y$

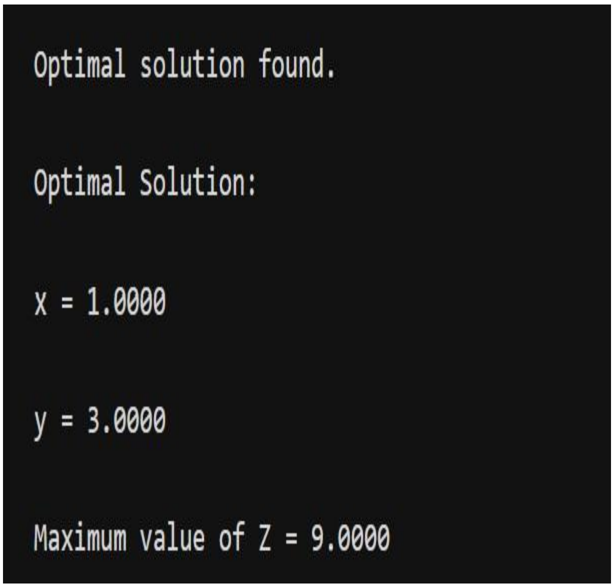
Subject to the constraints: $x + y \leq 4$; $2x + y \leq 5$; $x \geq 0, y \geq 0$

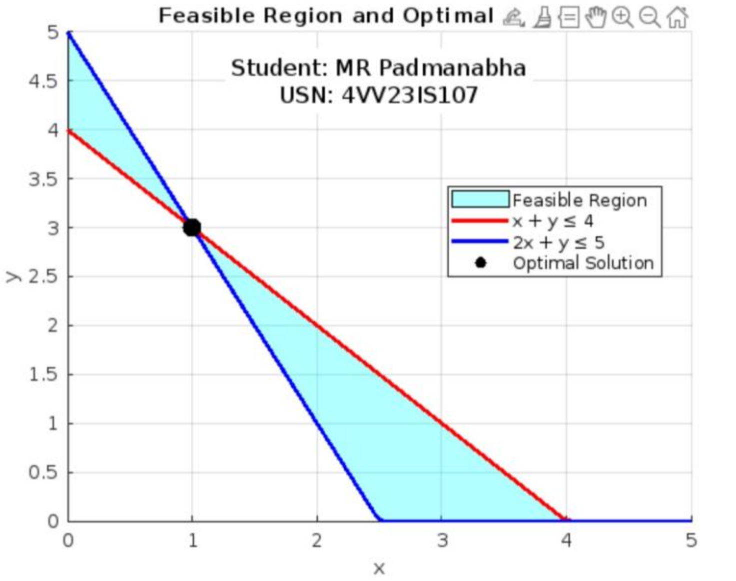
i. Write a MATLAB script to solve this **maximization problem** using the **Simplex method**.
ii. Display the optimal values of x and y, as well as the maximum value of the objective function Z.
iii. Use MATLAB's **linprog** function to verify your solution.

iv. Plot the feasible region and the optimal solution point on a 2D graph.

```
% Define the coefficients of the objective function (negated for maximization)
C = [-3 -2]; % Since linprog minimizes, we negate for maximization


% Define constraint coefficients and RHS values
A = [1 1; 2 1];  % Coefficients of inequality constraints
b = [4; 5];      % Right-hand side values
```

```matlab
% Define lower bounds (non-negative constraints)
lb = [0; 0];

% Solve using linprog (MATLAB's simplex method for LP problems)
[x_optimal, Z_max, exitflag, output] = linprog(C, A, b, [], [], lb);

% Convert objective function value to maximization format
Z_max = -Z_max;

% Display results
fprintf('Optimal Solution:\n');
fprintf('x = %.4f\n', x_optimal(1));
fprintf('y = %.4f\n', x_optimal(2));
fprintf('Maximum value of Z = %.4f\n', Z_max);

% Plot feasible region and optimal solution
x_vals = linspace(0, 5, 100);
y1 = max(0, 4 - x_vals); % First constraint
y2 = max(0, 5 - 2*x_vals); % Second constraint

figure;
hold on;
fill([x_vals fliplr(x_vals)], [y1 fliplr(y2)], 'cyan', 'FaceAlpha', 0.3); % Feasible
region
plot(x_vals, y1, 'r-', 'LineWidth', 2);
plot(x_vals, y2, 'b-', 'LineWidth', 2);
scatter(x_optimal(1), x_optimal(2), 100, 'ko', 'filled'); % Optimal solution point
xlabel('x');
ylabel('y');
title('Feasible Region and Optimal Solution');
legend('Feasible Region', 'x + y ≤ 4', '2x + y ≤ 5', 'Optimal Solution', 'Location',
'Best');
grid on;

% Define student details
student_name = 'MR Padmanabha';
usn = '4VV23IS107';

% Adjust placement of student details for correct visibility
annotation_text = sprintf('Student: %s\nUSN: %s', student_name, usn);
text(mean(x_vals), max(y1) + 0.5, annotation_text, 'FontSize', 12, 'Color', 'black',
'BackgroundColor', 'white', 'HorizontalAlignment', 'center');

hold off;
```





QUESTION 10:

Consider the following linear programming problem:

Minimize the objective function: $Z = 4x + 6y$

Subject to the constraints: $2x + y \geq 8\,;\;\; x + 2y \geq 6\,;\;\; x \geq 0, y \geq 0$

i. Write a MATLAB script to solve this minimization problem using linear programming (using the Simplex method).
ii. Display the optimal values of x and y, as well as the minimum value of the objective function Z.
iii. Use MATLAB's linprog function to solve the problem.
iv. Plot the feasible region and highlight the optimal solution point on a 2D graph.

```
c
% Define the coefficients of the objective function (minimize Z = 4x + 6y)
C = [4 6]; % Coefficients for minimization


% Define constraint coefficients and RHS values (converted to ≤ form for linprog)
A = [-2 -1; -1 -2];  % Convert "≥" constraints to "≤" by multiplying by -1
b = [-8; -6];        % Corresponding RHS values


% Define lower bounds (non-negative constraints)
lb = [0; 0];


% Solve using linprog (Simplex method for LP problems)
[x_optimal, Z_min, exitflag, output] = linprog(C, A, b, [], [], lb);


% Display results
fprintf('Optimal Solution:\n');
fprintf('x = %.4f\n', x_optimal(1));
fprintf('y = %.4f\n', x_optimal(2));
fprintf('Minimum value of Z = %.4f\n', Z_min);
```

```
% Plot feasible region and optimal solution
x_vals = linspace(0, 10, 100);
y1 = max(0, (8 - 2*x_vals)); % First constraint (converted)
y2 = max(0, (6 - x_vals)/2); % Second constraint (converted)

figure;
hold on;
fill([x_vals fliplr(x_vals)], [y1 fliplr(y2)], 'cyan', 'FaceAlpha', 0.3); % Feasible region
plot(x_vals, y1, 'r-', 'LineWidth', 2);
plot(x_vals, y2, 'b-', 'LineWidth', 2);
scatter(x_optimal(1), x_optimal(2), 100, 'ko', 'filled'); % Optimal solution point
xlabel('x');
ylabel('y');
title('Feasible Region and Optimal Solution');
legend('Feasible Region', '2x + y ≥ 8', 'x + 2y ≥ 6', 'Optimal Solution', 'Location', 'Best');
grid on;

% Define student details
student_name = 'MR Padmanabha';
usn = '4VV23IS107';

% Adjust placement of student details for correct visibility inside the graph
annotation_text = sprintf('Student: %s\nUSN: %s', student_name, usn);
text(x_optimal(1) + 0.5, x_optimal(2) + 0.5, annotation_text, 'FontSize', 14, 'FontWeight', 'normal', 'Color', 'black', 'BackgroundColor', 'white', 'HorizontalAlignment', 'left');
hold off;
```

```
Optimal solution found.

Optimal Solution:

x = 3.3333

y = 1.3333

Minimum value of Z = 21.3333
```