AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

Department of Mathematics
Amrita School of Engineering
Amrita Vishwa Vidyapeetham, Amaravati

Course: Discrete Mathematics

Course Code: 23MAT116

NAME:- P SAI LIKHITH
SECTION:- CSE - A
ROLL NUMBER :- AV.SC.U4CSE24306

VERIFIED BY:-

S.NO	TITLE	PAGE NO	SIGNATURE
1.	Write MatLab program to generate a truth table that consists of 3 statements: p, q, r.		
2.	Write recursive program for Fibonacci series in MatLab.		
3.	Implement the binary search as a recursive function in MatLab.		
4.	Write a MatLab program for permutation and combinations. Apply this implementation to the following problem. How many ways are there to select five players from a 10-member tennis team to make a trip to a match at another school?		
5.	Write a MatLab program to compute f_n for $n = 1, 2, \ldots, 10$. The recurrence for this question is $f(0) = 25$, $f_n = f(n-1) + 7 - \frac{7(n+1)}{n}$, $n \geq 2$.		
6.	Create a directed graph using an edge list, and then find the equivalent adjacency matrix representation of the graph.		
7.	Create a graph using an edge list, and then calculate the graph incidence matrix.		
8.	Create a directed graph using an edge list, and then calculate the incidence matrix.		
9.	Create and plot a graph, and then find the degree of each node.		
10.	Create and plot a directed graph. Calculate the shortest path between nodes.		
11.	Create and plot a graph with weighted edges. Find the shortest path between nodes, and specify two outputs to also return the length of the path.		

QUESTION 1:-Write MatLab program to generate a truth table that consists of 3 statements: *p*, *q*, *r*.

SOLUTION:-

```
% Truth Table Generator for 3 Variables: p, q, r

% Number of variables
n = 3;

% Generate all combinations of 0s and 1s
truthTable = dec2bin(0:2^n-1) - '0';  % Converts binary string to numeric array

% Extract p, q, r from the table
p = truthTable(:,1);
q = truthTable(:,2);
r = truthTable(:,3);

% Display the truth table
fprintf(' p   q   r\n');
fprintf('------------\n');
for i = 1:size(truthTable, 1)
    fprintf(' %d   %d   %d\n', p(i), q(i), r(i));
end
```

OUTPUT:-

```
 p   q   r
------------
 0   0   0
 0   0   1
 0   1   0
 0   1   1
 1   0   0
 1   0   1
 1   1   0
 1   1   1
```

QUESTION 2:-Write recursive program for Fibonacci series in MatLab.

SOLUTION:-

```
function f = fib(n)
    % Recursive function to compute the nth Fibonacci number
    if n == 0
        f = 0;
    elseif n == 1
        f = 1;
    else
        f = fib(n-1) + fib(n-2);
    end
end

% Display first N Fibonacci numbers
N = 10;  % You can change this value
fprintf('Fibonacci series up to term %d:\n', N);
for i = 0:N-1
    fprintf('%d ', fib(i));
end
fprintf('\n');
```

OUTPUT:-

```
Fibonacci series up to term 10:

0 1 1 2 3 5 8 13 21 34
```

QUESTION 3:- Implement the binary search as a recursive function in MatLab.

SOLUTION:-

```
function index = binarySearch(arr, target, low, high)
    % Recursive Binary Search
    % arr: sorted array
    % target: value to find
    % low, high: current bounds (use 1 and length(arr) initially)

    if low > high
        index = -1;  % Element not found
        return;
    end

    mid = floor((low + high) / 2);

    if arr(mid) == target
        index = mid;  % Element found
    elseif target < arr(mid)
        index = binarySearch(arr, target, low, mid - 1);  % Search left
    else
        index = binarySearch(arr, target, mid + 1, high);  % Search right
    end
end

% Sample sorted array
arr = [2, 4, 7, 10, 15, 20, 25];
target = 10;

% Call binary search
result = binarySearch(arr, target, 1, length(arr));

% Display result
if result == -1
    fprintf('Element %d not found.\n', target);
else
    fprintf('Element %d found at index %d.\n', target, result);
end
```

OUTPUT:-

```
Element 10 found at index 4.
```

QUESTION 4:- Write a MatLab program for permutation and combinations. Apply this implementation to the following problem. How many ways are there to select five players from a 10-member tennis team to make a trip to a match at another school?

SOLUTION:-

```
 function permutation_combination()
    clc;
    disp('Permutation and Combination Calculator');

    n = input('Enter the value of n (total items): ');
    r = input('Enter the value of r (selected items): ');

    % Permutation: nPr = n! / (n - r)!
    perm = factorial(n) / factorial(n - r);

    % Combination: nCr = n! / (r! * (n - r)!)
    comb = factorial(n) / (factorial(r) * factorial(n - r));

    fprintf('Permutation (P(%d,%d)) = %d\n', n, r, perm);
    fprintf('Combination (C(%d,%d)) = %d\n', n, r, comb);
end
n = 10;
r = 5;
ways = nchoosek(n, r);  % Built-in function for combinations
fprintf('Number of ways to choose 5 players from 10: %d\n', ways);
```

OUTPUT:-

```
Number of ways to choose 5 players from 10: 252
```

QUESTION 5:-

Write a MatLab program to compute f_n for $n = 1, 2, \ldots, 10$. The recurrence for this question is $f(0) = 25,\ f_n = f(n-1) + 7 - \dfrac{7(n+1)}{n}$, $n \geq 2$.

SOLUTION:-

```
 % MATLAB script to compute f_n for n = 1 to 10 using the recurrence relation

f = zeros(1, 11); % Preallocate for f(0) to f(10)
f(1) = 25;        % f(0) = 25 (index 1 in MATLAB due to 1-based indexing)

% Compute f(1) using custom logic since recurrence is defined for n >= 2
% Since f(1) is not defined in the recurrence, we need to define it explicitly
% Let's assume f(1) = f(0) + 7 - (7 * (1 + 1) / 1)
f(2) = f(1) + 7 - (7 * (2) / 1);  % f(1)

% Apply recurrence for n = 2 to 10
for n = 3:11
    f(n) = f(n - 1) + 7 - (7 * (n) / (n - 1));
end

% Display results
for n = 1:11
    fprintf('f(%d) = %.4f\n', n - 1, f(n));
end
```

OUTPUT:-

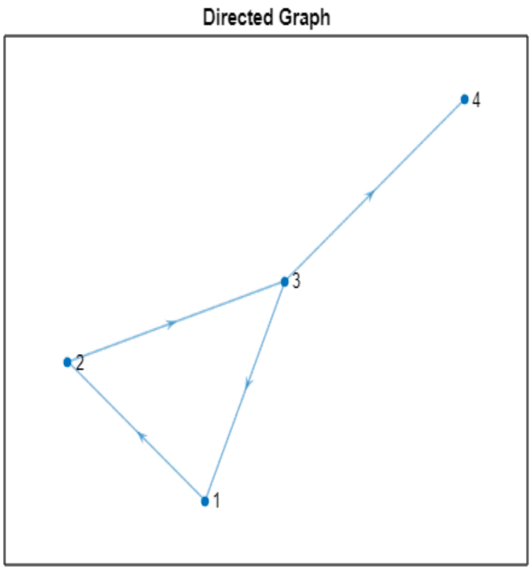
```
f(0) = 25.0000
f(1) = 18.0000
f(2) = 14.5000
f(3) = 12.1667
f(4) = 10.4167
f(5) = 9.0167
f(6) = 7.8500
f(7) = 6.8500
f(8) = 5.9750
f(9) = 5.1972
f(10) = 4.4972
```

QUESTION 6:-Create a directed graph using an edge list, and then find the equivalent adjacency matrix representation of the graph.

SOLUTION:-

```
 % Define edge list: each row represents a directed edge from source to target
% Example: edges = [1 2; 2 3; 3 1; 3 4]; means 1→2, 2→3, 3→1, 3→4
edges = [1 2; 2 3; 3 1; 3 4];

% Create directed graph
G = digraph(edges(:,1), edges(:,2));

% Plot the graph
figure;
plot(G);
title('Directed Graph');

% Get the adjacency matrix
adjMatrix = adjacency(G);

% Display adjacency matrix
disp('Adjacency Matrix:');
disp(full(adjMatrix));  % Use full to convert from sparse to regular matrix
```

OUTPUT:-



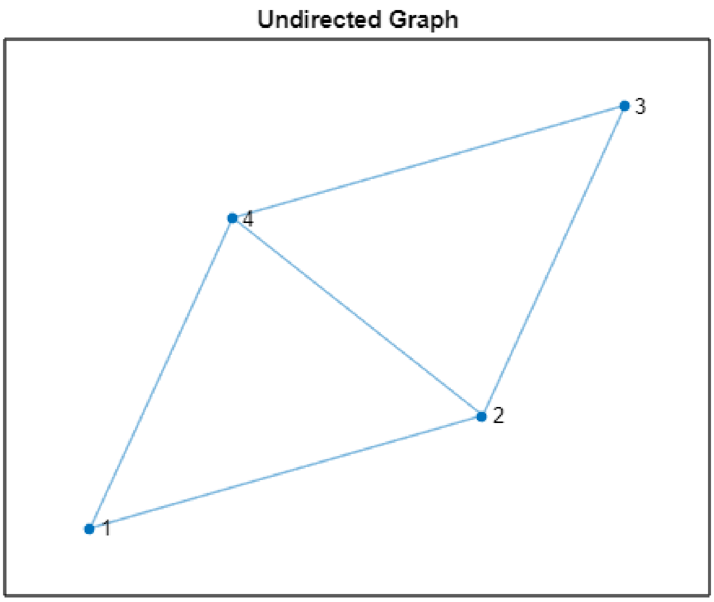

```
Adjacency Matrix:

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

QUESTION 7: Create a graph using an edge list, and then calculate the graph incidence matrix.

SOLUTION:-

```
  % Define the edge list (undirected graph example)
% Each row is an edge: [source, target]
edges = [1 2; 2 3; 3 4; 4 1; 2 4];

% Create an undirected graph
G = graph(edges(:,1), edges(:,2));

% Plot the graph
figure;
plot(G);
title('Undirected Graph');

% Compute incidence matrix
incMatrix = incidence(G);

% Display incidence matrix
disp('Incidence Matrix:');
disp(full(incMatrix));  % Convert from sparse to full matrix for readability
```

OUTPUT:-



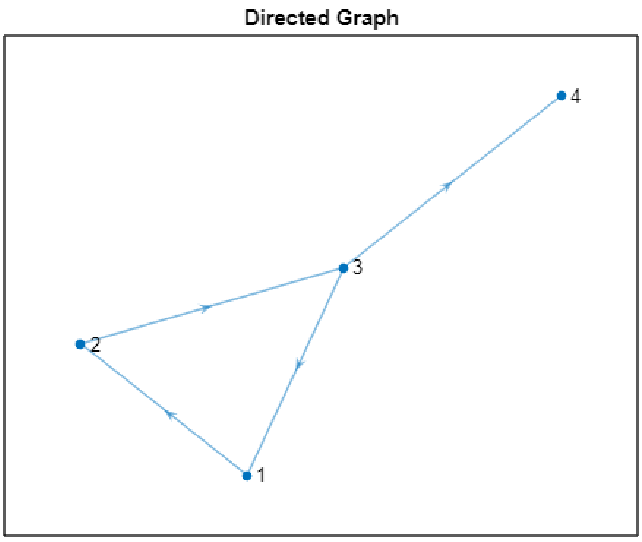

```
Incidence Matrix:
    -1    -1     0     0     0
     1     0    -1    -1     0
     0     0     1     0    -1
     0     1     0     1     1
```

QUESTION 8:-Create a directed graph using an edge list, and then calculate the incidence matrix.

SOLUTION:-

```
 % Define the edge list for a directed graph
% Each row: [source_node, target_node]
edges = [1 2; 2 3; 3 1; 3 4];

% Create the directed graph
G = digraph(edges(:,1), edges(:,2));

% Plot the graph
figure;
plot(G, 'Layout', 'force');
title('Directed Graph');

% Compute the incidence matrix
incMatrix = incidence(G);

% Display the incidence matrix
disp('Incidence Matrix of the Directed Graph:');
disp(full(incMatrix));  % Convert sparse matrix to full for easy viewing
```

OUTPUT:-

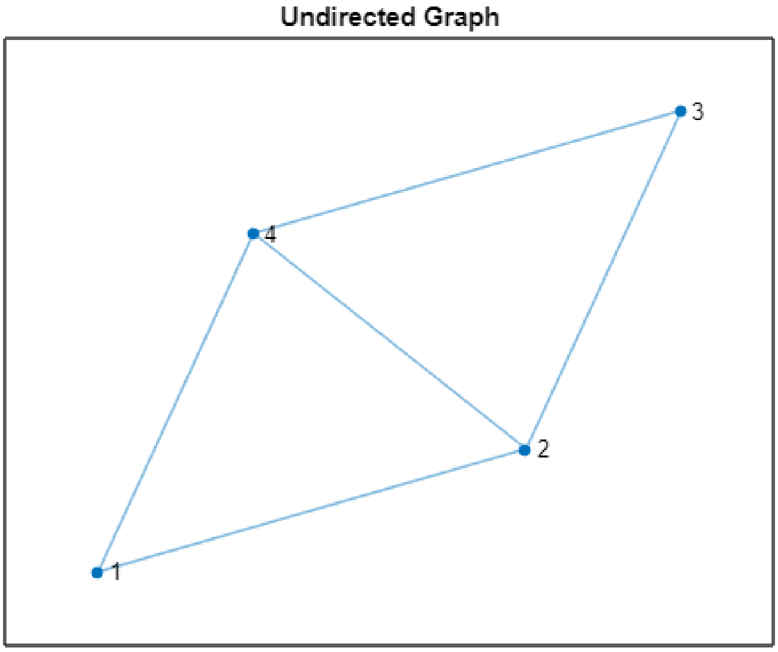



```
Incidence Matrix of the Directed Graph:
    -1     0     1     0
     1    -1     0     0
     0     1    -1    -1
     0     0     0     1
```

QUESTION 9:- Create and plot a graph, and then find the degree of each node.
SOLUTION:-

```
% Define edge list for an undirected graph
edges = [1 2; 2 3; 3 4; 4 1; 2 4];

% Create the graph
G = graph(edges(:,1), edges(:,2));

% Plot the graph
figure;
plot(G, 'Layout', 'force');
title('Undirected Graph');

% Compute node degrees
deg = degree(G);

% Display degrees
disp('Degree of each node:');
for i = 1:numel(deg)
    fprintf('Node %d: Degree = %d\n', i, deg(i));
end

%define edge list for an undirected graph
G = digraph(edges(:,1), edges(:,2));

% In-degree and Out-degree
inDeg = indegree(G);
outDeg = outdegree(G);

% Display results
disp('In-degree and Out-degree of each node:');
for i = 1:numnodes(G)
    fprintf('Node %d: In-degree = %d, Out-degree = %d\n', i, inDeg(i), outDeg(i));
end
```

OUTPUT:-




```
Degree of each node:
Node 1: Degree = 2
Node 2: Degree = 3
Node 3: Degree = 2
Node 4: Degree = 3
In-degree and Out-degree of each node:

Node 1: In-degree = 1, Out-degree = 1
Node 2: In-degree = 1, Out-degree = 2
Node 3: In-degree = 1, Out-degree = 1
Node 4: In-degree = 2, Out-degree = 1
```

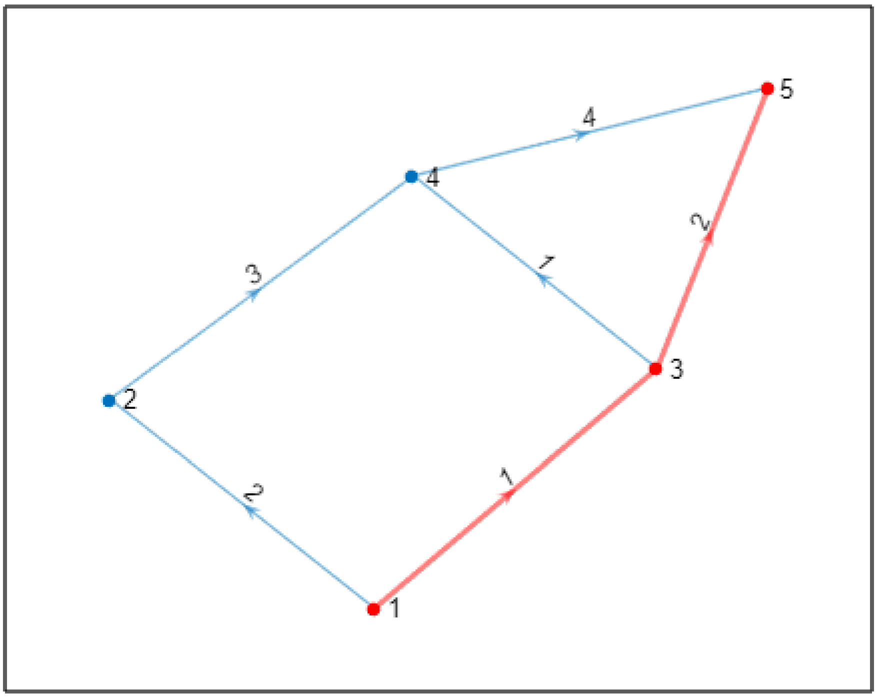
QUESTION 10:- Create and plot a directed graph. Calculate the shortest path between nodes.

SOLUTION:-

```
 % Define directed edge list: [source, target]
edges = [1 2; 1 3; 2 4; 3 4; 4 5; 3 5];

% Optionally define edge weights (if needed)
weights = [2 1 3 1 4 2];  % Example weights for each edge

% Create a directed graph with weights
G = digraph(edges(:,1), edges(:,2), weights);

% Plot the graph with edge weights labeled
figure;
p = plot(G, 'EdgeLabel', G.Edges.Weight, 'Layout', 'force');
title('Directed Graph with Edge Weights');

% Define start and end nodes for shortest path
startNode = 1;
endNode = 5;

% Find shortest path and its total distance
[path, totalWeight] = shortestpath(G, startNode, endNode);

% Highlight the shortest path in red
highlight(p, path, 'EdgeColor', 'r', 'LineWidth', 2);
highlight(p, path, 'NodeColor', 'r');

% Display result
fprintf('Shortest path from node %d to node %d: %s\n', ...
    startNode, endNode, mat2str(path));
fprintf('Total path weight: %d\n', totalWeight);
```

OUTPUT:-

Directed Graph with Edge Weights



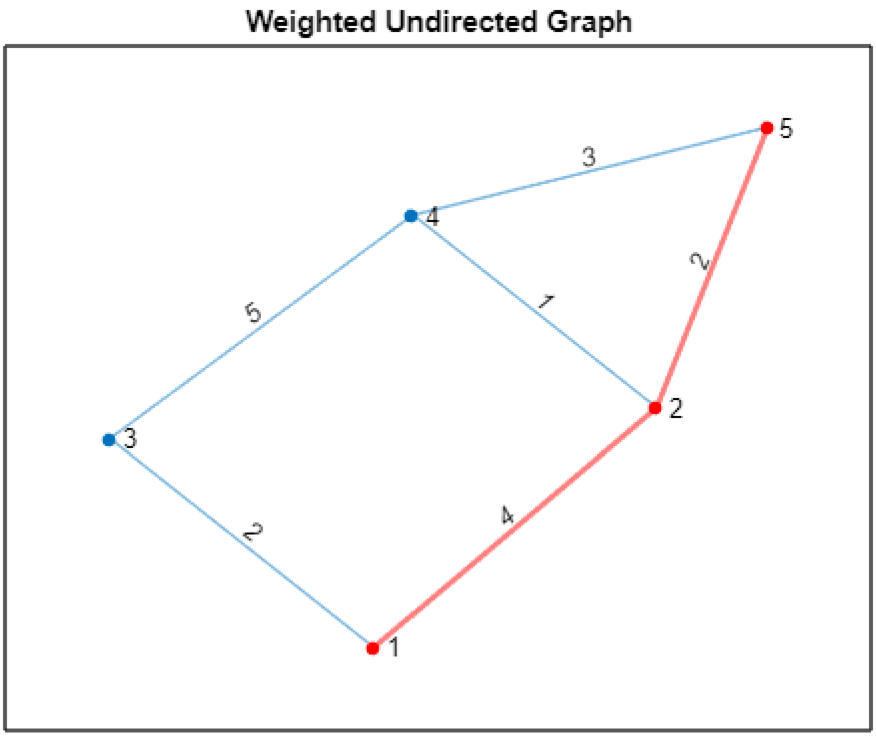

```
Shortest path from node 1 to node 5: [1 3 5]
Total path weight: 3
```

QUESTION 11:- Create and plot a graph with weighted edges. Find the shortest path between nodes, and specify two outputs to also return the length of the path.

SOLUTION:-

```
 % Define edge list: [node1, node2]
edges = [1 2; 1 3; 2 4; 3 4; 4 5; 2 5];

% Define weights for each edge
weights = [4 2 1 5 3 2];  % Corresponds to each edge above

% Create a weighted, undirected graph
G = graph(edges(:,1), edges(:,2), weights);

% Plot the graph with weights as edge labels
figure;
p = plot(G, 'EdgeLabel', G.Edges.Weight, 'Layout', 'force');
title('Weighted Undirected Graph');

% Specify start and end nodes for shortest path
startNode = 1;
endNode = 5;

% Find shortest path and path length
[path, totalLength] = shortestpath(G, startNode, endNode);

% Highlight the shortest path in red
highlight(p, path, 'EdgeColor', 'r', 'LineWidth', 2);
highlight(p, path, 'NodeColor', 'r');

% Display results
fprintf('Shortest path from node %d to node %d: %s\n', ...
    startNode, endNode, mat2str(path));
fprintf('Total path length: %d\n', totalLength);
```

OUTPUT:-




```
Shortest path from node 1 to node 5: [1 2 5]
Total path length: 6
```