Name : Abdul Rehman Section :B

Course : AI&ES Roll No : CT-22052

---------------------------------------------------------------------------------------------------------------------  
 **Task 1:**graph = {

    'A': ['B', 'E'],

    'B': ['F'],

    'C': ['G'],

    'D': ['E', 'H'],

    'E': ['A', 'D', 'H'],

    'F': ['B', 'G', 'I', 'J'],

    'G': ['C', 'F', 'J'],

    'H': ['D', 'E', 'I'],

    'I': ['F', 'H'],

    'J': ['F', 'G']

}

def bfs\_connected\_component(graph, start):

    # keep track of all visited nodes

    explored = []

    # keep track of nodes to be checked

    queue = [start]

    # keep looping until there are nodes still to be checked

    while queue:

        # pop shallowest node (first node) from queue

        node = queue.pop(0)

        if node not in explored:

            # add node to list of checked nodes

            explored.append(node)

            neighbours = graph[node]

            # add neighbours of node to queue

            for neighbour in neighbours:

                queue.append(neighbour)

    return explored

print(bfs\_connected\_component(graph, 'A'))

**Output:** **Task 2:**graph = {

    'A': ['B', 'C', 'E'],

    'B': ['A', 'D', 'E'],

    'C': ['A', 'F', 'G'],

    'D': ['B', 'E'],

    'E': ['A', 'B', 'D'],

    'F': ['C'],

    'G': ['C']

}

from collections import deque

def bfs\_shortest\_path(graph, start, end):

    queue = deque()

    queue.append((start, [start]))

    visited = set([start])

    while queue:

        node, path = queue.popleft()

        if node == end:

            return path

        for neighbor in graph[node]:

            if neighbor not in visited:

                visited.add(neighbor)

                queue.append((neighbor, path + [neighbor]))

print("Shortest path from G to D:", bfs\_shortest\_path(graph, 'G', 'D'))

**Output:** **Question # 1:**

**(a)**graph = {

    '1': ['2', '3', '4'],

    '2': ['1', '3', '4'],

    '3': ['1', '2', '4'],

    '4': ['1', '2', '3', '5'],

    '5': ['4', '6', '7', '8'],

    '6': ['5', '7', '8'],

    '7': ['5', '6', '8'],

    '8': ['5', '6', '7']

}

from collections import deque

def bfs(graph, start):

    queue = deque([start])

    visited = set([start])

    while queue:

        node = queue.popleft()

        print(node, end=" ")

        for neighbor in graph[node]:

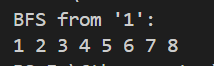
            if neighbor not in visited:

                visited.add(neighbor)

                queue.append(neighbor)

print("BFS from '1':")

bfs(graph, '1')

**Output:** **(b)**def find\_all\_paths(graph, start, end, path=[]):

    path = path + [start]

    if start == end:

        return [path]

    all\_paths = []

    for neighbor in graph[start]:

        if neighbor not in path:

            new\_paths = find\_all\_paths(graph, neighbor, end, path)

            for p in new\_paths:

                all\_paths.append(p)

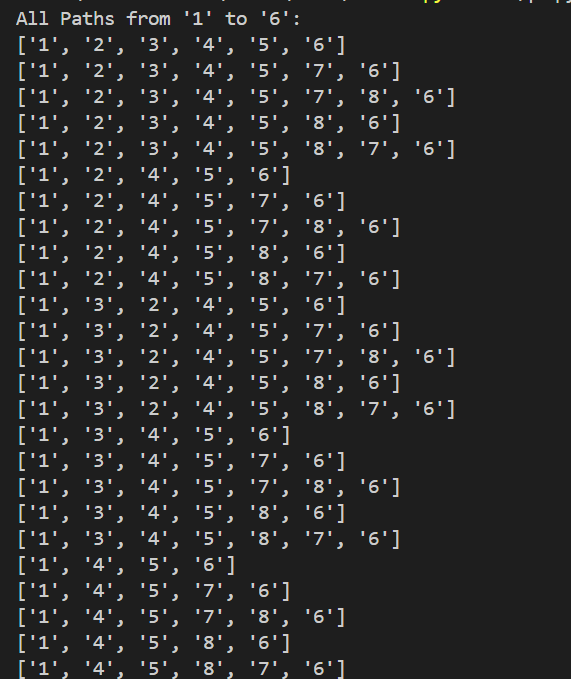
    return all\_paths

print("All Paths from '1' to '6':")

paths = find\_all\_paths(graph, '1', '6')

for p in paths:

    print(p)

**Output:**  
  
 (c)   
from collections import deque

def bfs\_shortest\_path(graph, start, end):

    queue = deque([(start, [start])])

    visited = set([start])

    while queue:

        node, path = queue.popleft()

        if node == end:

            return path

        for neighbor in graph[node]:

            if neighbor not in visited:

                visited.add(neighbor)

                queue.append((neighbor, path + [neighbor]))

print("Shortest Path from '1' to '6':", bfs\_shortest\_path(graph, '1', '6'))

**Output:** **Question # 2:**

**(a)|**graph = {

    'A': ['B', 'C', 'D'],

    'B': ['A', 'E'],

    'C': ['A', 'F'],

    'D': ['A', 'E', 'G'],

    'E': ['D', 'B', 'G'],

    'F': ['C', 'G'],

    'G': ['F', 'E', 'D'],

}

# part a

from collections import deque

def bfs(graph, start):

    queue = deque([start])

    visited = set([start])

    while queue:

        node = queue.popleft()

        print(node, end=" ")

        for neighbor in graph[node]:

            if neighbor not in visited:

                visited.add(neighbor)

                queue.append(neighbor)

print("BFS from 'A':")

bfs(graph, 'A')

**Output:**

**(b)**def find\_all\_paths(graph, start, end, path=[]):

    path = path + [start]

    if start == end:

        return [path]

    all\_paths = []

    for neighbor in graph[start]:

        if neighbor not in path:

            new\_paths = find\_all\_paths(graph, neighbor, end, path)

            for p in new\_paths:

                all\_paths.append(p)

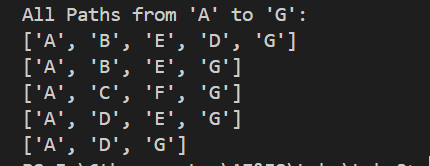
    return all\_paths

print("All Paths from 'A' to 'G':")

paths = find\_all\_paths(graph, 'A', 'G')

for p in paths:

    print(p)

**Output:** **(c)**def bfs\_shortest\_path(graph, start, end):

    queue = deque([(start, [start])])

    visited = set([start])

    while queue:

        node, path = queue.popleft()

        if node == end:

            return path

        for neighbor in graph[node]:

            if neighbor not in visited:

                visited.add(neighbor)

                queue.append((neighbor, path + [neighbor]))

print("Shortest Path from 'A' to 'G':", bfs\_shortest\_path(graph, 'A', 'G'))

**Output:  
  
**