Name of Student :- Pavan Ratan Gole

UID:- Na

Batch:- C4

Exp. No. 8

Aim:-

An Airline company is interested in airline routes among seven cities: Delhi, Mumbai, Jaipur, Pune, Bangalore, Ahmedabad, and Goa. It flies on the following ways:

a) Ahmedabad to Goa h) Jaipur to Ahmedabad
b) Ahmedabad to Pune i) Jaipur to Delhi
c) Bangalore to Jaipur j) Jaipur to Goa
d) Bangalore to Pune k) Mumbai to Pune
e) Delhi to Jaipur l) Pune to Bangalore
f) Delhi to Mumbai
g) Goa to Ahmedabad m) Pune to Mumbai

Questions:
a. Represent the above scenario using Graph. Represent the Graph in the Adjacency matrix and Adjacency List representation both
b. Design, apply and implement a strategy to find the route (direct or indirect) from one city to another city. (any route)
c. Is there any route from Delhi to Goa? (shortest route)

Program:-

```
import java.util.HashMap;

/**
* Graph
*/
class Queue
{



  public int front = -1;
  public  int rear = -1;
  public  Node arr[];
```

```java
int size = 7;

Queue(int kize)
{
    arr = new Node[kize];
    size = kize;
}

boolean isEmpty()
{
    if ((front == -1 && rear == -1))
    {
        return true;
    }
    return false;
}

boolean isFull()
{
    if ((size - 1 == rear && front == 0) || front - 1 == rear)
    {
        return true;
    }
    return false;
}

void enqueue(Node data)
{
    if (!isFull())
    {
        if (front == -1)
        {
            front = 0;
        }
        if (rear == size - 1)
        {
            rear = -1;
        }
        ++rear;
        arr[rear] = data;
    }
    else
    {
        System.out.println("Queue is FUll");
    }
}

void dequeue()
{
    if (!isEmpty())
    {

        if (rear == front)
        {
            arr[front] = null;
            rear = front = -1;
        }
        else
        {
            arr[front] = null;
            front++;
            if (front > size - 1)
            {
                front = 0;
            }
        }
    }
    else
    {
        System.out.println("Queue is Empty");
    }
}

Node peek()
{
    if (!isEmpty())
    {
        return arr[front];
    }
    return null;
}

Node getlast()
{
    if (!isEmpty())
    {
        return arr[rear];
    }

    return null;
}
};


class Node{
int data;
int source;
Node next;
Node(int data) {
    this.data = data;
}
}

public class Graph {
int size = 7;
Node[] graph = new Node[size];
Node tail = null;
int[] visited = new int[7];
int[] parent = {-1,-1,-1,-1,-1,-1,-1};
int[] distance = new int[7];


int[] bfs(int source) {
    Queue queue = new Queue(7);
    queue.enqueue(graph[source]);
    visited[source] = 1;
    while (!queue.isEmpty()) {
        Node node = queue.peek();
        Node temp = node;
```

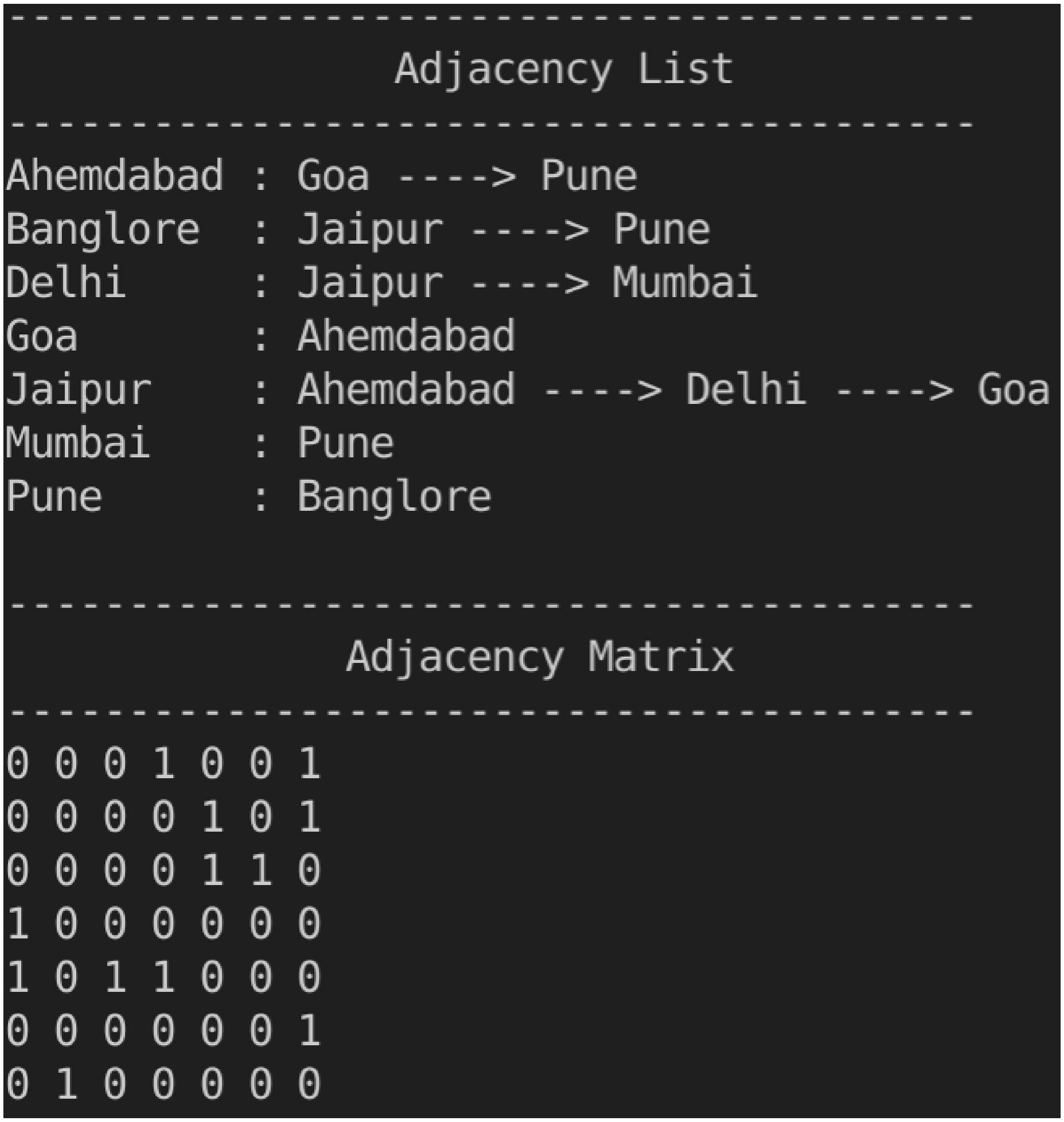
```java
                while(temp != null) {
                    if (this.visited[temp.data] == 0) {
                        this.parent[temp.data] = node.source;
                        this.visited[temp.data] = 1;
                        queue.enqueue(graph[temp.data]);
                    }
                    temp = temp.next;
                }
                queue.dequeue();
            }
            return parent;
        }


        //Return the shortest path
        void path(int source,int destination) {
            bfs(source);
            String s  = ""+ destination;
            while(destination != source) {
                s+=parent[destination];
                destination = parent[destination];
            }

            for (int i = s.length() - 1; i >= 0; i--) {
                if(i != 0)
                    System.out.print(decode(s.charAt(i))  + " -----> ");
                else {
                    System.out.print(decode(s.charAt(i)));
                }
            }
            System.out.println();

        }


        //Add edge to Adjacency List
        void addedge(int source , int destination) {
            Node node = new Node(destination);
            node.source = source;
            if(graph[source] == null) {
```

```
            graph[source] = tail = node;
        }
        else {
            tail.next = node;
            tail = node;
        }
    }

    //Display Adjacency List
    void display() {
        for (int i = 0; i < graph.length; i++) {
            System.out.print(String.format("%-9s : ", decode((char) (i +
'0'))));
            Node temp = graph[i];
            while(temp  != null) {
               if(temp.next != null) {
                   System.out.print(decode( (char) (temp.data + '0')) + "
----> ");
               }
               else {
                System.out.print(decode( (char) (temp.data + '0')));
               }
               temp = temp.next;
            }
            System.out.println();

        }
        System.out.println();
    }

    //Display Adjacency Matrix
    void disadm() {
        int adj_matrix[][] = {
            {0,0,0,1,0,0,1},
            {0,0,0,0,1,0,1},
            {0,0,0,0,1,1,0},
            {1,0,0,0,0,0,0},
            {1,0,1,1,0,0,0},
            {0,0,0,0,0,0,1},
            {0,1,0,0,0,0,0}
```

```java
        };

        for (int i = 0; i < adj_matrix.length; i++) {
            for (int j = 0; j < adj_matrix.length; j++) {
                System.out.print(adj_matrix[i][j] + " ");
            }
            System.out.println();
        }
        System.out.println();
    }

    //Convert Number to proper city name
    String decode(char data) {
        String[] arr =
{"Ahemdabad","Banglore","Delhi","Goa","Jaipur","Mumbai","Pune"};
        return arr[Integer.parseInt("" + data)];
    }

    //Convert city name to proper city Number
    int encode(String data) {
        HashMap<String,Integer> map = new HashMap<>();
        map.put("Ahemdabad", 0);
        map.put("Banglore",1);
        map.put("Delhi",2);
        map.put("Goa",3);
        map.put("Jaipur",4);
        map.put("Mumbai",5);
        map.put("Pune",6);
       return map.get(data);
    }

    public static void main(String[] args) {
        Graph graph = new Graph();
        //Ahemdabad
        //Banglore
        //Delhi
        //Goa
        //Jaipur
        //Mumbai
        //Pune
        graph.addedge(0, 3);
        graph.addedge(0, 6);
        graph.addedge(1, 4);
        graph.addedge(1, 6);
        graph.addedge(2, 4);
        graph.addedge(2, 5);
        graph.addedge(3, 0);
        graph.addedge(4, 0);
        graph.addedge(4,2);
        graph.addedge(4, 3);
        graph.addedge(5, 6);
        graph.addedge(6, 1);
        System.out.println("-----------------------------------------");
        System.out.println(String.format("%30s","Adjacency List"));
        System.out.println("-----------------------------------------");
        graph.display();
        System.out.println("-----------------------------------------");
        System.out.println(String.format("%30s","Adjacency Matrix"));
        System.out.println("-----------------------------------------");
        graph.disadm();
        System.out.println("-----------------------------------------");
        System.out.println(String.format("%30s","Shortest Route"));
        System.out.println("-----------------------------------------");
        graph.path(graph.encode("Delhi"),graph.encode("Goa"));
        System.out.println();
    }
}
```

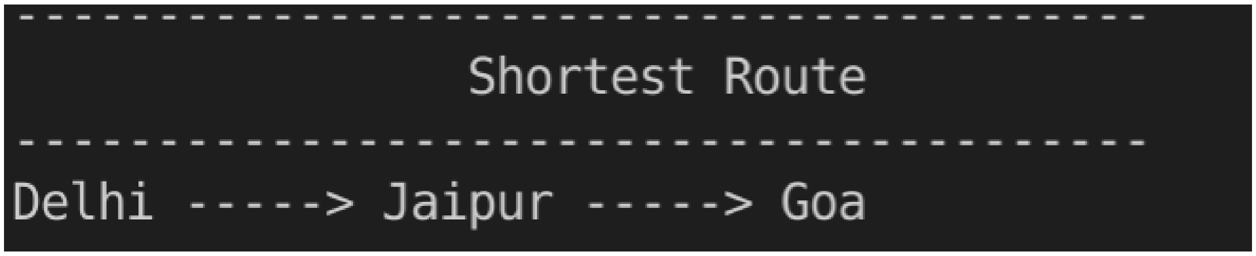
Output:-

1)



```
----------------------------------------
                Adjacency List
----------------------------------------
Ahemdabad : Goa ----> Pune
Banglore  : Jaipur ----> Pune
Delhi     : Jaipur ----> Mumbai
Goa       : Ahemdabad
Jaipur    : Ahemdabad ----> Delhi ----> Goa
Mumbai    : Pune
Pune      : Banglore


----------------------------------------
              Adjacency Matrix
----------------------------------------
0 0 0 1 0 0 1
0 0 0 0 1 0 1
0 0 0 0 1 1 0
1 0 0 0 0 0 0
1 0 1 1 0 0 0
0 0 0 0 0 0 1
0 1 0 0 0 0 0
```

2)



```
----------------------------------------
               Shortest Route
----------------------------------------
Delhi -----> Jaipur -----> Goa
```