```java


/* BFS SSSP Single Source Shortest Path on an un-weighted graph
import java.util.*;

class shippingRoutes_SSSP_BFS{
    static Node[] G;
    static int E;
    static int N;
    static int P;

    static class Node {
        List<Edge> adj;
        String name;
        int n;
        public boolean visited;
        int layer;
        public Node(int N){
            adj = new ArrayList<Edge>();
            n=N;
            layer = -1;
            visited = false;
            name = null;
        }
    }
    static class Edge{
        int to, weight;
        public Edge(int t, int w){
            to=t;
            weight = w;
        }
    }
    public static void makeGraph(int n){
    G = new Node[n];
        for(int i =0; i<n; i++){
            G[i]=new Node(i);
        }
    }
    public static void addEdge(int u,int v, int w){
        G[u].adj.add(new Edge(v,w));
        G[v].adj.add(new Edge(u,w));
    }
    public static int charN(char c){
        return c;
    }
    public static void init_bfs(){
        for(int i = 0; i<G.length; i++){
            G[i].layer = -1;
            G[i].visited = false;
        }
    }
    public static int bfs(int s, int t){
        G[s].layer = 0;
        Queue<Integer> q = new LinkedList<Integer>();
        q.add(s);
        int cur = -1;
        while (!q.isEmpty()){
            cur = q.peek();
            G[cur].visited = true;
            for (Edge e : G[cur].adj){
                if(!(G[e.to].visited)){
                    q.add(G[e.to].n);
                    G[e.to].visited = true;
                    G[e.to].layer = G[cur].layer + 1;
                }
            }
            q.poll();
        }
//      for(Node node : G)
//      System.out.print(node.layer+" ");
        return G[t].layer;
    }


    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.println("SHIPPING ROUTES OUTPUT");
        System.out.println();
        int K = scan.nextInt();
        String temp = "String";
        String a = "String";
        String b = "String";
        int size = 0;
        int length = 0;
        for(int k =0; k<K; k++){
            N = scan.nextInt();
            E = scan.nextInt();
            P = scan.nextInt();
            makeGraph(N);
            System.out.println("DATA SET  " + (k+1));
            System.out.println();
            for(int i = 0; i<N; i++){
                temp = scan.next();
                G[i].name = temp;
            }
            for(int i = 0; i<E; i++){
                a = scan.next();
                b = scan.next();
                for(int x = 0; x<N; x++){
                    for(int y = 0; y<N; y++){
                        if(G[x].name.equalsIgnoreCase(a) &&
G[y].name.equalsIgnoreCase(b))
                        {
                            addEdge(x,y,1);
                        }
                    }
                }
            }
            for(int i = 0; i<P; i++){
                size = scan.nextInt();
                a = scan.next();
                b = scan.next();
                for(int x = 0; x<N; x++){
                    for(int y = 0; y<N; y++){
                        if(G[x].name.equalsIgnoreCase(a) &&
G[y].name.equalsIgnoreCase(b))
                        {
                            init_bfs();
                            length = bfs(x,y);
                        }
                    }
                }
                if(length>0)
                System.out.println("$" + length*size*100);
                else
                    System.out.println("NO SHIPMENT POSSIBLE");
            }
            System.out.println();

        }
        System.out.println("END OF OUTPUT");
        scan.close();
    }
}
```