```java

/*BFS Breadth First Search to find the number of layers of connectivity between a
 * source node and all the other nodes in the graph.
 * Thus it can also be used to find Shortest paths for Single-Source Shortest Path
 * problems (SSSP)
 * */

import java.util.*;

class SingleSource_ShortestPath_BFS{
    static Node[] G;
    static int M;
    static int N;
    static int cnt = 0;
    static int conCompCnt=0;
    static Stack<Integer> s = new Stack<Integer>();

    static class Node {
        List<Edge> adj;
        int n;
        public boolean visited;
        int layer;
        public Node(int N){
            adj = new ArrayList<Edge>();
            n=N;
            layer = -1;
            visited = false;
        }
    }
    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];
```

```
39          for(int i =0; i<n; i++){
40              G[i]=new Node(i);
41          }
42      }
43      public static void addEdge(int u,int v, int w){
44          G[u].adj.add(new Edge(v,w));
45          G[v].adj.add(new Edge(u,w));
46      }
47      public static int charN(char c){
48          return c;
49      }
50      public static int bfs(int s, int t){
51          G[s].layer = 0;
52          Queue<Integer> q = new LinkedList<Integer>();
53          q.add(s);
54          int cur = -1;
55          while (!q.isEmpty()){
56              cur = q.peek();
57              G[cur].visited = true;
58              for (Edge e : G[cur].adj){
59                  if(!(G[e.to].visited)){
60                      q.add(G[e.to].n);
61                      G[e.to].visited = true;
62                      G[e.to].layer = G[cur].layer + 1;
63                  }
64              }
65              q.poll();
66          }
67          for(Node node : G)
68          System.out.print(node.layer+" ");
69          return G[t].layer;
70      }
71      public static void main(String[] args){
72          Scanner scan = new Scanner(System.in);
73
74          int K = Integer.parseInt(scan.nextLine());
75          int u = -1;
76          int v = -1;
77          String temp = "String";
78          for(int k =0; k<K; k++){
```

```
79              temp = scan.nextLine();
80              N = charN(temp.charAt(0))-65+1;
81              System.out.println(N);
82              makeGraph(N);
83              while((temp = scan.nextLine())!=null){
84                  if(temp.isEmpty()){
85                      break;
86                  }
87
88                   u = charN(temp.charAt(0))-65;
89                   v = charN(temp.charAt(1))-65;
90                   addEdge(u,v,1);
91              }
92              bfs(0, 5);
93          }
94          scan.close();
95      }
96  }
97
```