```java

/*  DFS Depth Fist Search to find Connected Components of a graph


import java.util.*;

class conCompDFS{
    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;

        public Node(int N){
            adj = new ArrayList<Edge>();
            n=N;
            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];
        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 dfs(int n){

        if(G[n].visited){
            return;
        }

        G[n].visited = true;
        s.push(n);

        cnt++;

        for(Edge e : G[n].adj)
        {
            dfs(e.to);
        }

    }
    public static int conComp(Node[] g){
            for(int i = 0; i<g.length;i++){
                if(!(g[i].visited)){
                    conCompCnt++;
                    dfs(i);
                }
            }
        return conCompCnt;
    }

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);

        int K = Integer.parseInt(scan.nextLine());
        int u = -1;
        int v = -1;
//      String t = scan.nextLine();
//      String q = scan.nextLine();

        String temp = "String";
        for(int k =1; k<K; k++){
            temp = scan.nextLine();
            N = charN(temp.charAt(0))-65+1;
//          System.out.println(N);
            makeGraph(N);

            while((temp = scan.nextLine())!=null){
                if(temp.isEmpty()){
                    break;
                }

                 u = charN(temp.charAt(0))-65;
                 v = charN(temp.charAt(1))-65;
//                          System.out.println(u);
//                          System.out.println(v);

                 addEdge(u,v,1);

//              System.out.println(k);
            }
            conComp(G);
            if(k>1)
                System.out.println();
            System.out.println(conCompCnt);
            conCompCnt = 0;



//          for(int z =0; z<N; z++){
//              System.out.println(G[z].adj.size());
//          }
//          System.out.println(s);
```

```

            scan.close();
        }
    }
}

```