CuriousFleas.java

```java
import java.io.*;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.PriorityQueue;

public class CuriousFleas implements Comparable<CuriousFleas> {

    private static int[] xAxis = {1, 0, -1, 0};
    private static int[] yAxis = {0, 1, 0, -1};
    private ArrayList<Integer> fleas;
    private int die, diePos;
    private int fleasTaken, depth;

    private CuriousFleas(ArrayList<Integer> fleas, int die, int diePos, int fleasTaken) {
        this.fleas = fleas;
        this.die = die;
        this.diePos = diePos;
        this.fleasTaken = fleasTaken;
        depth = 0;
    }

    private CuriousFleas(CuriousFleas curiousFleas) {
        fleas = new ArrayList<>( curiousFleas.fleas );
        die = curiousFleas.die;
        diePos = curiousFleas.diePos;
        fleasTaken = curiousFleas.fleasTaken;
        depth = curiousFleas.depth;
    }

    @Override
    public int compareTo( CuriousFleas cf ){
        return depth + fleas.size() - ( cf.fleas.size() + cf.depth );
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        CuriousFleas that = (CuriousFleas) o;

        if (die != that.die) return false;
        if (diePos != that.diePos) return false;
        if (fleasTaken != that.fleasTaken) return false;
        return fleas != null ? fleas.equals(that.fleas) : that.fleas == null;
    }

    @Override
    public int hashCode() {
        int result = fleas != null ? fleas.hashCode() : 0;
        result = 31 * result + die;
        result = 31 * result + diePos;
        result = 31 * result + fleasTaken;
        return result;
    }

    void updateMap(){
        ArrayList<Integer> fleas2 = new ArrayList<>();
        int side = 7 - ( die & 7 );
        int aux = (int) Math.pow( 2, side - 1 );
```

```
        if ( ( ( fleasTaken >>> ( side - 1) ) & 1 ) != 0) {
            fleas2.add(diePos);
            fleasTaken -= aux;
        }

        for (int tmp : fleas) {
            if (tmp == diePos)
                fleasTaken += aux;
            else
                fleas2.add( tmp );
        }
        fleas = fleas2;
    }

    void move(int xDir, int yDir){
        if ( xDir == -1 ){
            int side2 = die & 7;
            int side1 = 7 - ( (die >>> 3) & 7 );
            int side3 = die >>> 6;
            die = side1 + 8 * side2 + 64 * side3;
        }
        else if ( xDir == 1 ){
            int side2 = 7 - ( die & 7 );
            int side1 = (die >>> 3) & 7;
            int side3 = die >>> 6;
            die = side1 + 8 * side2 + 64 * side3;
        }
        else if ( yDir == 1 ){
            int side1 = 7 - (die >>> 6);
            int side2 = (die >>> 3) & 7;
            int side3 = die & 7;
            die = side1 + 8 * side2 + 64 * side3;
        }
        else if ( yDir == -1 ){
            int side1 = die >>> 6;
            int side2 = (die >>> 3) & 7;
            int side3 = 7 - ( die & 7 );
            die = side1 + 8 * side2 + 64 * side3;
        }
        depth++;
        updateMap();
    }

    static void printMap(CuriousFleas s){
        char[][] map = new char[4][4];
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++)
                map[i][j] = '.';

        for (int tmp : s.fleas){
            int a = tmp >>> 2;
            int b = tmp & 3;
            map[a][b] = '#';
        }
        int a = s.diePos >>> 2;
        int b = s.diePos & 3;
        map[a][b] = 'D';

        System.out.println( 7 - (s.die & 7) );
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                System.out.print(map[i][j]);
            }
            System.out.println();
        }
```

```
        System.out.println();
    }

    static int search(CuriousFleas fleas){
        HashSet<CuriousFleas> hashSet = new HashSet<>();
        PriorityQueue<CuriousFleas> priorityQueue = new PriorityQueue<>();
        priorityQueue.add(fleas);
        while ( !priorityQueue.isEmpty() ){
            CuriousFleas u = priorityQueue.poll();
            if ( u.fleas.size() == 0)
                return u.depth;
            //CuriousFleas.printMap(u);
            int x = u.diePos >>> 2;
            int y = u.diePos & 3;
            int a, b;
            for (int i = 0; i < 4; i++){
                a = x + xAxis[i];
                b = y + yAxis[i];
                if ( a < 4 && a >= 0 && b < 4 && b >= 0){
                    CuriousFleas cf = new CuriousFleas( u );
                    cf.diePos = a * 4 + b;
                    cf.move( xAxis[i], yAxis[i] );
                    if ( hashSet.add(cf) )
                        priorityQueue.add(cf);
                }
            }
        }
        return -1;
    }

    public static void main(String[] args) throws IOException{
        //BufferedReader br = new BufferedReader ( new InputStreamReader ( System.in ) );
        BufferedReader br = new BufferedReader( new FileReader(
              new File( "/home/juan/Documents/Maratones/Rogue/src/input.txt" ) ) );
        int T = Integer.parseInt( br.readLine() );
        br.readLine();
        char[][] map = new char[4][4];
        ArrayList<Integer> fleas = new ArrayList<>(6);
        int diePos = 0, die = 1 + 8 * 2 + 64 * 3;
        for (int c = 0; c < T; c++){
            for (int i = 0; i < 4; i++)
                map[i] = br.readLine().toCharArray();
            br.readLine();

            fleas.clear();
            for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 4; j++) {
                    if (map[i][j] == 'X')
                        fleas.add(i * 4 + j);
                    else if (map[i][j] == 'D')
                        diePos = i * 4 + j;
                }
            }

            CuriousFleas curiousFleas = new CuriousFleas(fleas, die, diePos, 0);

            int r = CuriousFleas.search(curiousFleas);
            if ( r != -1 )
                System.out.println(r);
            else
                System.out.println("impossible");
        }
    }
}
```