```
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;


public class Cubifier {


	public static void main(String[] args) throws IOException {


		FileReader fr = null;

		String cubeFile = "rubikstest.txt"; //CHANGE THIS FILE TO CHANGE WHICH CUBE YOU'RE
SOLVING


		BufferedReader br = null;


		fr = new FileReader(cubeFile);

		br = new BufferedReader(fr);


		String line = br.readLine();

		br.close();


		char[] inputArray = line.toCharArray();


		//TriedCubes listOfTries = new TriedCubes();

		Cube cube = new Cube(inputArray);
```

```
        System.out.println("Your original cube:");

        cube.PrintCube(cube);


        if (cube.Solve(cube)){


            //if solution was found

            System.out.println("A solution was found! Your cube's solution path is above.");

            System.out.println("The solution path is printed with the next move above the
last as you scroll up, with the solved cube at the top.");


        } else {


            //if no solution was found

            System.out.println("Sorry, but we couldn't solve your cube.");

            System.out.println("This was most likely an error on our end. Please send us a
message about it!");

            System.out.println("Here is the unsolved cube you wound up with.");

            cube.PrintCube(cube);

        }


    }


}
```

```
public enum Move {


    F("F"),

    B("B"),

    L("L"),

    R("R"),

    U("U"),

    D("D");


    private final String move;


    Move(String move) {

        this.move = move;

  }


  public String getMove() {

        return this.move;

  }


}
```

```
public class Square {

        //properties
        //private int _position;
        private Color _color;

        //constructor
        public Square(char colour) {

                //this._position = position;
                if (colour == 'r'){
                        this._color = Color.RED;
                }
                else if (colour == 'b'){
                        this._color = Color.BLUE;
                }
                else if (colour == 'w'){
                        this._color = Color.WHITE;
                }
                else if (colour == 'o'){
                        this._color = Color.ORANGE;
                }
                else if (colour == 'y'){
                        this._color = Color.YELLOW;
                }
        else if (colour == 'g'){

            this._color = Color.GREEN;

        }

        else {

            System.out.println("An error occured. Color of square could not be assigned.");

        }


    }


    //getter

    //public int getPosition(){ return this._position; }

    public Color getColor(){ return this._color; }


    //setter

    //public void setPosition(int pos){ this._position = pos; }


}
```

```
public enum Color {


       RED("R"),

       BLUE("B"),

       WHITE("W"),

       ORANGE("O"),

       YELLOW("Y"),

       GREEN("G");


       private final String color;


       Color(String color) {

         this.color = color;

     }


     public String getColor() {

         return this.color;

     }


}
```

```
import java.util.ArrayList;

import java.util.List;


public class Cube {


        //properties

        private Square[] _cube = new Square[24];

        //public static List<Square[]> _tried = new ArrayList<Square[]>();

        TriedCubes listOfTries = new TriedCubes();


        //constructor

        public Cube(char[] inputArray){


                for (int counter = 0; counter < 24; counter++) {


                        Square temp = new Square(inputArray[counter]);

                        this._cube[counter] = temp;


                }


        }




        public boolean Solve(Cube cube){


                cube.PrintCube(cube);
```

```
boolean solved = false;

for (int counter = 0; counter < listOfTries.getLength(); counter++){

        if (cube._cube == listOfTries.getList(counter)){

                //if this cube layout has been seen before, do nothing.
                return false;

        }

}

//now that we know this hasn't been tried yet, we're trying it
//TriedCubes._tried.add(cube._cube);
listOfTries.Add(cube._cube);

if (CubeSolved(cube)){

        PrintCube(cube);
        return true;

} else {

        //if the cube isn't solved yet:
```

```
        if (Solve(moveF(cube))){ //TRY FRONT

                //if it's solved when i do move F
                solved = true;

        } else if (Solve(moveL(cube))) { //TRY LEFT

                //if it's solved when i do move L
                solved = true;

        } else if (Solve(moveU(cube))) { //TRY TOP/UPPER

                //if it's solved when i do move U
                solved = true;

        }

}

if (solved){

        PrintCube(cube);
        return true;

} else {
```

```
                    return false;

                }

            }

            public void PrintCube(Cube cube){

                //top

                System.out.println("");

                System.out.println("        " + cube._cube[16].getColor() + "" +
cube._cube[17].getColor());

                System.out.println("        " + cube._cube[18].getColor() + "" +
cube._cube[19].getColor());

                //left, front, right, back

                System.out.println(cube._cube[12].getColor() + "" + cube._cube[13].getColor() + " " +
cube._cube[0].getColor() + "" + cube._cube[1].getColor() + " " + cube._cube[4].getColor() + "" +
cube._cube[5].getColor() + " " + cube._cube[8].getColor() + "" + cube._cube[9].getColor());

                System.out.println(cube._cube[14].getColor() + "" + cube._cube[15].getColor() + " " +
cube._cube[2].getColor() + "" + cube._cube[3].getColor() + " " + cube._cube[6].getColor() + "" +
cube._cube[7].getColor() + " " + cube._cube[10].getColor() + "" + cube._cube[11].getColor());

                //lower

                System.out.println("        " + cube._cube[20].getColor() + "" +
cube._cube[21].getColor());

                System.out.println("        " + cube._cube[22].getColor() + "" +
cube._cube[23].getColor());
```

```
    }

    public boolean CubeSolved(Cube cube) {

        //checks if the cube is solved
        boolean solved = false;

        if(cube._cube[0] == cube._cube[1] && cube._cube[0] == cube._cube[2] &&
cube._cube[0] == cube._cube[3]){
            if(cube._cube[4] == cube._cube[5] && cube._cube[4] == cube._cube[6] &&
cube._cube[4] == cube._cube[7]){
                if(cube._cube[8] == cube._cube[9] && cube._cube[8] == cube._cube[10]
&& cube._cube[8] == cube._cube[11]){
                    if(cube._cube[12] == cube._cube[13] && cube._cube[12] ==
cube._cube[14] && cube._cube[12] == cube._cube[15]){
                        if(cube._cube[16] == cube._cube[17] && cube._cube[16]
== cube._cube[18] && cube._cube[16] == cube._cube[19]){
                            if(cube._cube[20] == cube._cube[21] &&
cube._cube[20] == cube._cube[22] && cube._cube[20] == cube._cube[23]){
                                //if cube is solved
                                solved = true;
                            }
                        }
                    }
                }
            }
        }
        return solved;

}

public Cube moveF(Cube cube){

        Square temp;
        Square temp2;
        Square temp3;
        Square temp4;

        //front face
        temp = cube._cube[1];
        cube._cube[1] = cube._cube[0]; //moves 0 to 1
        temp2 = cube._cube[3];
        cube._cube[3] = temp;    //moves 1 to 3
        temp = cube._cube[2];
        cube._cube[2] = temp2; //moves 3 to 2
        cube._cube[0] = temp;    //moves 2 to 0

        //left to top
        temp = cube._cube[18];
        temp2 = cube._cube[19];
        cube._cube[18] = cube._cube[15];    //moves 15 to 18
        cube._cube[19] = cube._cube[13];    //moves 13 to 19
        //top to right

        temp3 = cube._cube[4];

        temp4 = cube._cube[6];

        cube._cube[4] = temp;    //moves 18 to 4

        cube._cube[6] = temp2; //moves 19 to 6


        //right to bottom

        temp = cube._cube[21];

        temp2 = cube._cube[20];

        cube._cube[21] = temp3; //moves 4 to 21

        cube._cube[20] = temp4; //moves 6 to 20


        //bottom to left

        cube._cube[15] = temp;    //moves 21 to 15

        cube._cube[13] = temp2; //moves 20 to 13


        return cube;


}


public Cube moveL(Cube cube){


        Square temp;

        Square temp2;

        Square temp3;

        Square temp4;
```

```
//left face

temp = cube._cube[13];

cube._cube[13] = cube._cube[12]; //moves 12 to 13

temp2 = cube._cube[15];

cube._cube[15] = temp;     //moves 13 to 15

temp = cube._cube[14];

cube._cube[14] = temp2; //moves 15 to 14

cube._cube[12] = temp;     //moves 14 to 12


//back to top

temp = cube._cube[16];

temp2 = cube._cube[18];

cube._cube[16] = cube._cube[11];

cube._cube[18] = cube._cube[9];


//top to front

temp3 = cube._cube[0];

temp4 = cube._cube[2];

cube._cube[0] = temp;

cube._cube[2] = temp2;


//front to bottom

temp = cube._cube[20];

temp2 = cube._cube[22];

cube._cube[20] = temp3;
```

```
        cube._cube[22] = temp4;

        //bottom to back
        cube._cube[11] = temp;
        cube._cube[9] = temp2;
        return cube;

}

public Cube moveU(Cube cube){

        Square temp;
        Square temp2;
        Square temp3;
        Square temp4;

        //top face
        temp = cube._cube[17];
        this._cube[17] = cube._cube[16]; //moves 16 to 17
        temp2 = cube._cube[19];
        cube._cube[19] = temp;    //moves 17 to 19
        temp = cube._cube[18];
        cube._cube[18] = temp2; //moves 19 to 18
        cube._cube[16] = temp;    //moves 18 to 16

        //left to back
        temp = cube._cube[9];

        temp2 = cube._cube[8];

        cube._cube[9] = cube._cube[13];

        cube._cube[8] = cube._cube[12];


        //back to right

        temp3 = cube._cube[5];

        temp4 = cube._cube[4];

        cube._cube[5] = temp;

        cube._cube[4] = temp2;


        //right to front

        temp = cube._cube[1];

        temp2 = cube._cube[0];

        cube._cube[1] = temp3;

        cube._cube[0] = temp4;


        //front to left

        cube._cube[13] = temp;

        cube._cube[12] = temp2;


        return cube;


}
```

obgywboyyyrgrborgowrwgwb