Indian Institute of technology, Guwahati
Department of Computer Science and Engineering
Data Structure Lab: (CS210)
Offline Assignment: 9

Date: 23rd October 2017. **Total Marks: 40**

Deadline: 10PM, 29th October 2017. (Hard Deadline)

1. You are given *N* colorful cubes, each having a distinct weight. Cubes are not monochromatic – indeed, every face of a cube is colored with a different color. Your job is to build the tallest possible tower of cubes subject to the restrictions that (1) we never put a heavier cube on a lighter one, and (2) the bottom face of every cube (except the bottom one) must have the same color as the top face of the cube below it. [20]

   Input
   The input may contain several test cases. The first line of each test case contains an integer *N* ($1 \le N \le 500$) indicating the number of cubes you are given. The *i*th of the next *N* lines contains the description of the *i*th cube. A cube is described by giving the colors of its faces in the following order: front, back, left, right, top, and bottom face. For your convenience colors are identified by integers in the range 1 to 100. You may assume that cubes are given in increasing order of their weights; that is, cube 1 is the lightest and cube *N* is the heaviest. The input terminates with a value 0 for *N*.

   Output
   For each case, start by printing the test case number on its own line as shown in the sample output. On the next line, print the number of cubes in the tallest possible tower. The next line describes the cubes in your tower from top to bottom with one description per line. Each description gives the serial number of this cube in the input, followed by a single whitespace character and then the identification string (front, back, left, right, top, or bottom of the top face of the cube in the tower. There may be multiple solutions, but any one of them is acceptable. Print a blank line between two successive test cases.

   Sample Input

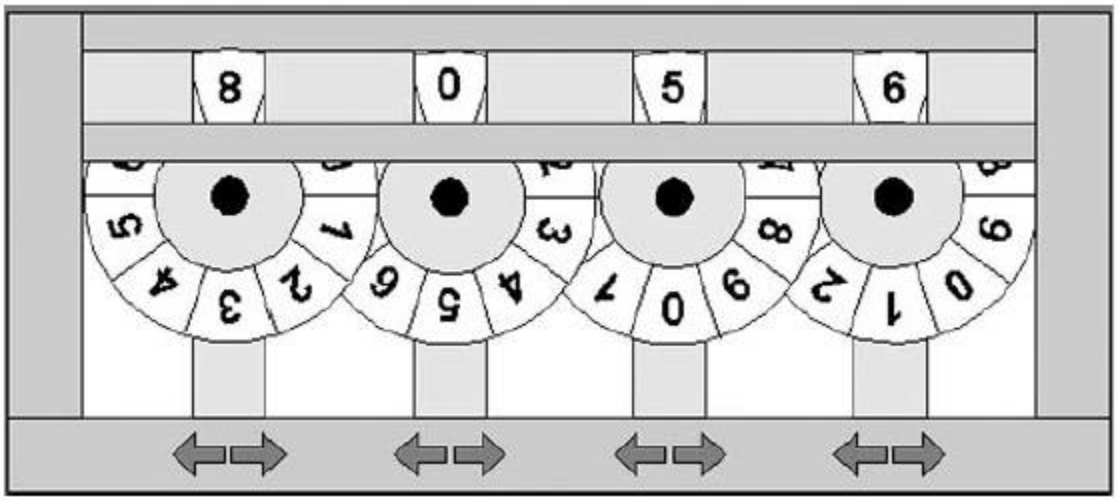
```
3
1 2 2 2 1 2
3 3 3 3 3 3
3 2 1 1 1 1
10
1 5 10 3 6 5
2 6 7 3 6 9
5 7 3 2 1 9
1 3 3 5 8 10
6 6 2 2 4 4
1 2 3 4 5 6
10 9 8 7 6 5
6 1 2 3 4 7
1 2 3 3 2 1
3 2 1 1 2 3
0
```

Sample Output

```
Case #1
2
2 front
3 front

Case #2
8
1 bottom
2 back
3 right
4 left
6 top
8 front
9 front
10 top
```

2. Consider the following mathematical machine. Digits ranging from 0 to 9 are printed consecutively (clockwise) on the periphery of each wheel. The topmost digits of the wheels form a four-digit integer. For example, in the following figure the wheels form the integer 8,056. Each wheel has two buttons associated with it. Pressing the button marked with a left arrow rotates the wheel one digit in the clockwise direction and pressing the one marked with the right arrow rotates it by one digit in the opposite direction. [20]



We start with an initial configuration of the wheels, with the topmost digits forming the integer S1S2S3S4. You will be given a set of n forbidden configurations Fi1Fi2Fi3Fi4 ($1 \le i \le n$) and a target configuration T1T2T3T4. Your job is to write a program to calculate the minimum number of button presses required to transform the initial configuration to the target configuration without passing through a forbidden one.

Input

The first line of the input contains an integer *N* giving the number of test cases. A blank line then follows. The first line of each test case contains the initial configuration of the wheels, specified by four digits. Two consecutive digits are separated by a space. The next line contains the target

configuration. The third line contains an integer *n* giving the number of forbidden configurations. Each of the following *n* lines contains a forbidden configuration.
There is a blank line between two consecutive input sets.

Output
For each test case in the input print a line containing the minimum number of button presses required. If the target configuration is not reachable print “-1”.

Sample Input

```
2
8 0 5 6
6 5 0 8
5
8 0 5 7
8 0 4 7
5 5 0 8
7 5 0 8
6 4 0 8
0 0 0 0
5 3 1 7
8
0 0 0 1
0 0 0 9
0 0 1 0
0 0 9 0
0 1 0 0
0 9 0 0
1 0 0 0
9 0 0 0
```

Sample Output

```
14
-1
```