Wheels!

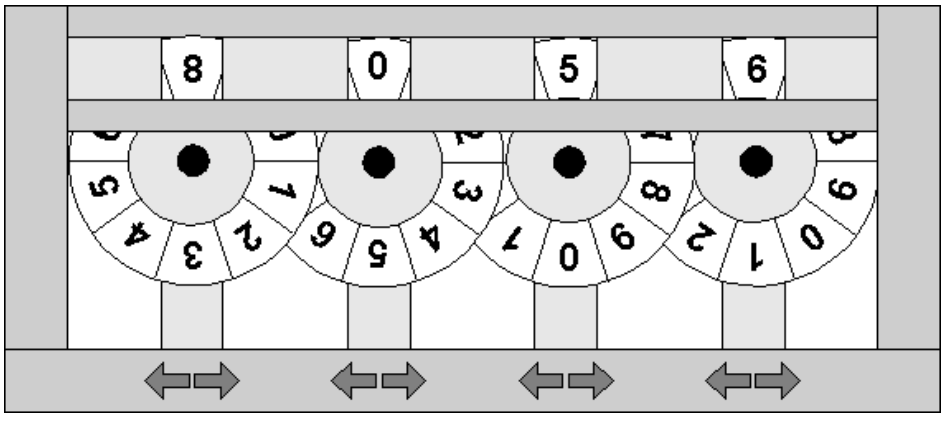
This assignment is due March 11th, 2004, 11:59pm (Pacific Time). Do not wait until the last minute to start this assignment.

Rules

- Programs must compile with g++ under Linux
- You are expected to split your program in several .cc and .h files and to use makefiles
- Programs should be written in a structured an understandable way
- The grade will be based on correctness, comments, style, readability, and code efficiency
- The programs will be inspected for plagiarism using automatic tools
- At the beginning of each source code file you should include your full name with upper-case LAST name, assignment number, SSN, login, class section, lab section
- Submit a README file along with the source code
- Programming assignments have to be submitted electronically
- No late programming assignment will be accepted

Description

In this problem we will be considering a game played with four wheels. 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 figure above the wheels form the integer 8056. 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.



The game starts with an initial configuration of the wheels. Say, in the initial configuration the topmost digits form the integer $S_1S_2S_3S_4$. You will be given some (say, n) forbidden configurations $F_{i,1}F_{i,2}F_{i,3}F_{i,4}$ ($1 \le i \le n$) and a target configuration $T_1T_2T_3T_4$. Your job will be to write a program that can calculate the minimum number of button presses required to transform the initial configuration to the target configuration by never passing through a forbidden one.

Input

The first line of the input contains an integer N giving the number of test cases to follow. The first line of each test case contains the initial configuration of the wheels specified by 4 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. You should also print one of the possible sequences of key presses and the intermediate configurations. If the target configuration is not reachable, then print that the target configuration is not possible.

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

Input # :1
8 0 5 6
7 0 5 6
6 0 5 6
6 1 5 6
6 2 5 6
6 3 5 6
6 4 5 6
6 5 5 6
6 5 6 6
6 5 7 6
6 5 8 6
6 5 9 6
6 5 0 6
6 5 0 7
6 5 0 8
Minimum keypresses required :14
Input # :2
There is no way to reach from source to destination !!!

Program Interface

The program reads the input from a file which format is specified in earlier paragraph. The program must accept the name of the input and output files on the command-line as follows

wheels <inputfile> <outputfile>

where the <inputfile> is the actual name of the input file and <outputfile> is the name of the output file.

Coding and Testing

In order to facilitate the design/code/debug cycle, you should take care to develop the program in an incremental fashion. If you try to write the whole program at once, you probably will not get a completely working program. Design, develop, and test so that you have a working program at each step. Build a program by adding working and tested pieces.

We encourage using STL for this project (if you think it is needed). Report your choices of data structures in the README file.

To make sure that your program is working correctly, you can compare your output with our implementation of "wheels". The executable can be downloaded from the project homepage.

Report

Submit a README file (plain text) that gives an overview of your program. Explain the implementation choices you made. Describe the strengths and weaknesses of your implementation.

Grading

This assignment is worth 100 points, divided as follows:

Description	*points*
Correctness	45
Robustness	20
Error-free compilation, program style (class design/implementation, program design, README)	35

Extra Credit

An optional extra credit (worth 25 points) to this project is posted on the project homepage.

Acknowledgements

This assignment is based on a problem for the ACM International Collegiate Programming Contest