Sean the Cuber

Time limit: 15 seconds

Sean is one of the best cubers in China. Today, he is participating in the Fewest Moves Count event of the Chinese Competition of Pocket Cube (CCPC).

A pocket cube is the $2 \times 2 \times 2$ equivalent of the Rubik's cube. Each face of the cube can be rotated, either clockwise or counterclockwise. A standard solved pocket cube has the following color scheme (as shown in Figure 1): the top face is yellow, the bottom face is white, the left face is orange, the right face is red, the front face is blue, the back face is green.

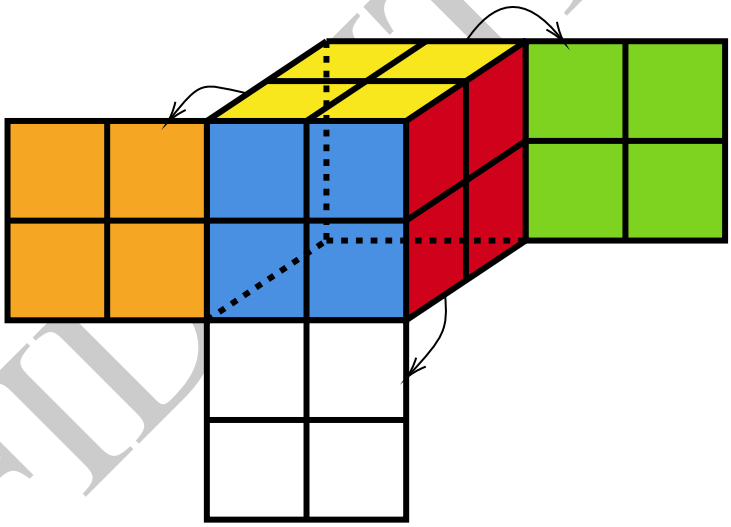

Figure 1: The solved state of a standard pocket cube

The Fewest Moves Count event requires a player to find a sequence of moves to transform one configuration of the pocket cube into another, and the player who finds the shortest such sequence among all participants wins the event. In this event, the player can twist any of the six faces in either direction; also, rotating the entire pocket cube is allowed. Note that a quarter twist counts as one move, and a half twist counts as two moves; however, rotating the entire cube doesn't count into the total number of moves.

Since there are millions of essentially different configurations, the task is too hard for Sean and he can't solve it independently. He wants you to write a program to find the minimum number of moves, given the initial and final configurations. Can you help him?

Input

The first line of input contains a single integer T ($1 \leq T \leq 250\,000$), denoting the number of the test cases.

Each test case begins with an empty line followed by six lines, representing the initial and final configurations. A configuration is represented as a net of the pocket cube in the following format:

```
  XX
  XX
XXXXXXXX
XXXXXXXX
  XX
  XX
```

where `X` is one of the uppercase letters in {`G`, `O`, `Y`, `R`, `W`, `B`}, denoting green, orange, yellow, red, white, and blue, respectively, and all other characters are whitespaces. The initial and final configurations are horizontally stacked together, with each line split by a vertical bar `|`. In other words, the 1st to the 8th columns represent the initial

configuration, the 9th column is vertical bars, and the 10th to the 17th columns represent the final configuration. It is guaranteed that both configurations are legal, that is, they can be recovered to the solved state by a finite sequence of moves.

Output

For each test case, output a single integer in a line, denoting the minimum number of moves required to transform the initial configuration into the final one.

Sample Input 1	Sample Output 1

2
 	
  GO | WG
  GO | WY
OOYBYWBW|ROGBRYRG
RRYRBWRB|BOGBWYBW
  GY | YO
  WG | RO
 	
  WR | YR
  OR | OW
OYGBWGYB|OWBRBYBG
BWGWRBYY|YOWRYGWO
  OG | GG
  OR | BR
	2
	11

Note

In the first case of the sample data, an optimal move sequence is shown in Figure 2. The total number of moves is 2 (note that the first step does not count into the number of moves), and it can be shown that it can't be done in fewer than 2 moves.

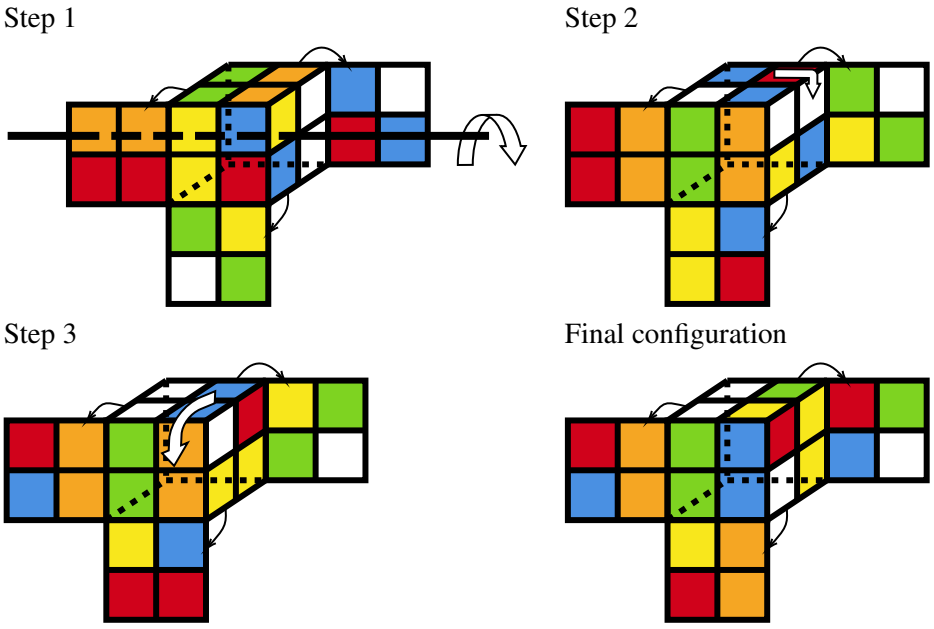

Figure 2: Optimal move sequence for the first case