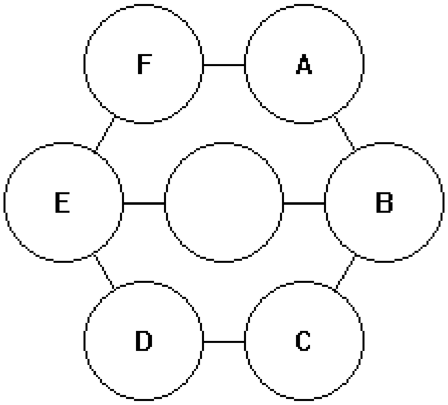
I • Theta Puzzle

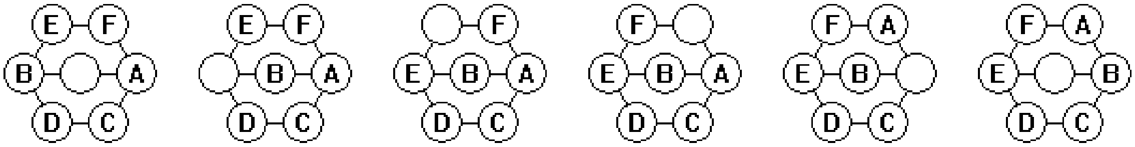
The *Theta Puzzle* consists of a base with 6 positions at the vertices of a regular hexagon and another position at the center, connected as shown in the figure below. There are six tokens labeled **A**, **B**, **C**, **D**, **E** and **F**. A single move of the puzzle is to move a token to an adjacent empty position (along an allowed connection – the line segments in the diagram below). The idea of the puzzle is to start with an initial arrangement of tokens with the center empty and, by a sequence of moves, get to the configuration in the figure below.



An initial position for the puzzle is given by a permutation of the letters **A** through **F**. The first letter starts at **A** in the figure, the next at **B** and so on.

A sequence of moves is specified by listing the labels of tokens to be moved, in the order they are to be moved.

For example, to solve the puzzle **FACDBE**, use the moves **BEFAB**.



Note: Not all starting permutations can be solved.

Write a program which, given an initial permutation, either finds the *shortest* sequence of moves to solve the puzzle or determines that there is no solution.

Input

The first line of input contains a single integer ***P***, (1 ≤ ***P*** ≤ 1000), which is the number of data sets that follow. Each data set is a single line that contains the data set number, followed by a space, followed by a permutation of the letters **A** through **F** giving the initial puzzle position.

Output

For each data set there is a single line of output. If there is no solution, the line contains a decimal integer giving the data set number followed by a single space, followed by the string **NO SOLUTION**. If there is a solution, the line contains a decimal integer giving the data set number followed by a single space, followed by the number of moves in the solution, followed by a single space, followed by the solution as a string of the letters **A** through **F**. If the number of moves is zero (0), you should still output the space after the 0, even though there is no string of letters.

Sample Input	Sample Output
`12`	`1 5 BEFAB`
`1 FACDBE`	`2 0`
`2 ABCDEF`	`3 19 DABFECABFEDBACDEFAB`
`3 ADCEFB`	`4 NO SOLUTION`
`4 ADCEBF`	`5 29 BCDEBCAFBCAFBCEDFAECBAFDCBAFE`
`5 FEDCBA`	`6 NO SOLUTION`
`6 FEDCAB`	`7 19 CBFACBFACDEFACDEFAB`
`7 ECBFAD`	`8 NO SOLUTION`
`8 ECBFDA`	`9 13 CDAFBEDCBEDCB`
`9 DCEBFA`	`10 NO SOLUTION`
`10 DCEBAF`	`11 21 DAEBDAEBDCFEBDCABEFAB`
`11 CBEADF`	`12 16 FAEDBCAFBCAFEDCB`
`12 BDEAFC`	