3272 cNteSahruPfefrlefe

Preston Digitation is a magician who specializes in card tricks. One thing Preston cannot get just right is perfect in-shuffles. A perfect in-shuffle is one where a deck of 52 cards is divided in half and then the two halves are perfectly interleaved so that the top card of the lower half of the deck becomes the top card of the shuffled deck. If we number the cards 0 (top) to 51 (bottom), the resulting deck after a perfect in-shuffle will look like the following:

26 0 27 1 28 2 29 3 30 4 31 5 32 6 ... 51 25

Preston finds that he makes at most one mistake per shuffle. For example, cards 2 and 28 might end up interchanged, resulting in a shuffled deck that looks like this:

26 0 27 1 2 28 29 3 30 4 31 5 32 6 ... 51 25

These exchanges of two adjacent cards are the only mistakes Preston makes. After one shuffle, it is easy for him to see if and where he has made a mistake, but after several shuffles this becomes increasingly difficult. He would like you to write a program that can determine his mistakes (if any).

Input

Input will consist of multiple problem instances. The first line will be a single integer indicating the number of problem instances. Each problem instance will consist of a single line containing the cards of a deck which has been shuffled between 1 and 10 times. All decks will be of size 52.

Note: The sample input below shows multiple lines for a problem instance. The actual input data for a problem instance is contained on a single line.

Output

For each problem instance, output the case number (starting at 1), followed by the number of shuffles that were used for that instance. If there were no mistakes made during the shuffling, output the line

`No error in any shuffle`

Otherwise, output a set of lines of the form

`Error in shuffle` n `at location` m

where n is a shuffle where an error occurred and m is the location of the error. Shuffles are numbered starting with 1 and the location value should indicate the first location of the two cards that were swapped in that shuffle (where the top of the deck is position 0). In the example described above, the cards in positions 4 and 5 (the cards numbered 2 and 28) are incorrect, so m would be 4 in this case. List all errors in order of increasing n. If one or more shuffles have no errors, do not print any line for them. If there are multiple solutions, pick the solution with the fewest number of errors. I there are still several solutions, pick the lexicographically first (regarding the error positions).

Print a blank line after each dataset.

Sample Input

```
3
26 0 27 1 2 28 29 3 30 4 31 5 32 6 33 7 34
8 35 9 36 10 37 11 38 12 39 13 40 14 41 15
42 16 43 17 44 18 45 19 46 20 47 21 48 22
49 23 50 24 51 25
26 0 27 1 28 2 29 3 30 4 31 5 32 6 33 7 34
8 35 9 36 10 37 11 38 12 39 13 40 14 41 15
42 16 43 17 44 18 45 19 46 20 47 21 48 22
49 23 50 24 51 25
49 26 43 40 37 34 31 28 25 22 19 16 13 10
7 4 1 51 48 45 42 39 36 33 24 27 30 21 18
15 12 9 6 3 0 50 47 44 41 38 35 32 29 46
23 20 17 2 11 8 5 14
```

Sample Output

```
Case 1
Number of shuffles = 1
Error in shuffle 1 at location 4

Case 2
Number of shuffles = 1
No error in any shuffle

Case 3
Number of shuffles = 9
Error in shuffle 3 at location 3
Error in shuffle 7 at location 11
Error in shuffle 8 at location 38
```