Problem A: Train Swapping

Input file: `swap.in`
Output file: `swap.out`

At an old railway station, you may still encounter one of the last remaining "train swappers". A train swapper is an employee of the railroad, whose sole job it is to rearrange the carriages of trains.

Once the carriages are arranged in the optimal order, all the train driver has to do is drop the carriages off, one by one, at the stations for which the load is meant.

The title "train swapper" stems from the first person who performed this task, at a station close to a railway bridge. Instead of opening up vertically, the bridge rotated around a piller in the center of the river. After rotating the bridge 90 degrees, boats could pass left or right. The first train swapper had discovered that the bridge could be operated with *at most two* carriages on it. By rotating the bridge 180 degrees, the carriages switched place, allowing him to rearrange the carriages (as a side effect, the carriages then faced the opposite directions, but train carriages can move either way, so who cares).

Now that almost all train swappers have died out, the railway company would like to automate their operation. Part of the program to be developed is a routine which decides, for a given train, the least number of swaps of two adjacent carriages necessary to order the train. Your assignment is to create a routine that computes the minimal number of swaps.

Questions

1. For a given train T, let $m[T]$ denote the minimal number of swaps to order train T. What is the *largest* value of $m[T]$ over all trains T with $L > 0$ carriages?
2. Write a program that satisfies the specification below.

Input specification

The input contains on the first line the number of test cases (N). Each test case consists of two input lines. The first line of a test case contains an integer L, determining the length of the train ($0 <= L <= 50$). The second line of a test case contains a permutation of the numbers 1 through L, indicating the current order of the carriages. The carriages should be ordered such that carriage 1 comes first, then 2, etc., with carriage L coming last.

Output specification

For each test case output the sentence: "`Optimal train swapping takes` S `swaps.`" where S is an integer representing the minimal number of swaps to order the train.

Sample input

```
3
3
1 3 2
4
4 3 2 1
2
2 1
```

Sample output

```
Optimal train swapping takes 1 swaps.
Optimal train swapping takes 6 swaps.
Optimal train swapping takes 1 swaps.
```

Problem B: Safebreaker

Input file: `safe.in`
Output file: `safe.out`

We are observing someone playing a game similar to Mastermind (TM). The object of this game is to find a secret code by intelligent guesswork, assisted by some clues. In this case the secret code is a 4-digit number in the inclusive range from 0000 to 9999, say "3321". The player makes a first random guess, say "1223" and then, as for each of the future guesses, gets a clue telling how close the guess is. A clue consists of two numbers: the number of correct digits (in this case, one: the "2" at the third position) and the additional number of digits guessed correctly but in the wrong place (in this case, two: the "1" and the "2"). The clue would in this case be: "1/2". For the guess "1110", the clue would be "0/1", since there are no correct digits and only one misplaced digit. (Notice that there is only one digit 1 misplaced.)

Write a program that, given a set of guesses and corresponding clues, tries to find the secret code.

Input specification

The first line of input specifies the number of test cases (*N*) your program has to process. Each test case consists of a first line containing the number of guesses *G* (0 <= *G* <= 10), and *G* subsequent lines consisting of exactly 8 characters: a code of four digits, a blank, a digit indicating the number of correct digits, a "/", and a digit indicating the number of correct but misplaced digits.

Output specification

For each test case, the output contains a single line saying either:

`impossible` if there is no code consistent with all guesses.

n, where *n* is the secret code, if there is exactly one code consistent with all guesses.

`indeterminate` if there is more than one code which is consistent with all guesses.

Sample input

```
4
6
9793 0/1
2384 0/2
6264 0/1
3383 1/0
2795 0/0
0218 1/0
1
1234 4/0
```

```
1
1234 2/2
2
6428 3/0
1357 3/0
```

Sample output

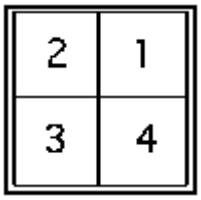
```
3411
1234
indeterminate
impossible
```

Problem C: Quadtrees

Input file: `quad.in`
Output file: `quad.out`

A modern computer artist works with black-and-white images of 32 x 32 units, for a total of 1024 pixels per image. One of the operations the artist performs is "adding" two images together, to form a new image. In the resulting image a pixel is black if it was black in at least one of the component images, otherwise it is white.

A quadtree is a representation format used to encode images. The fundamental idea behind the quadtree is that any image can be split into four quadrants. Each quadrant may again be split in four subquadrants, etc. In the quadtree, the image is represented by a parent node, while the four quadrants are represented by four child nodes, in a predetermined order as shown in the figure below.

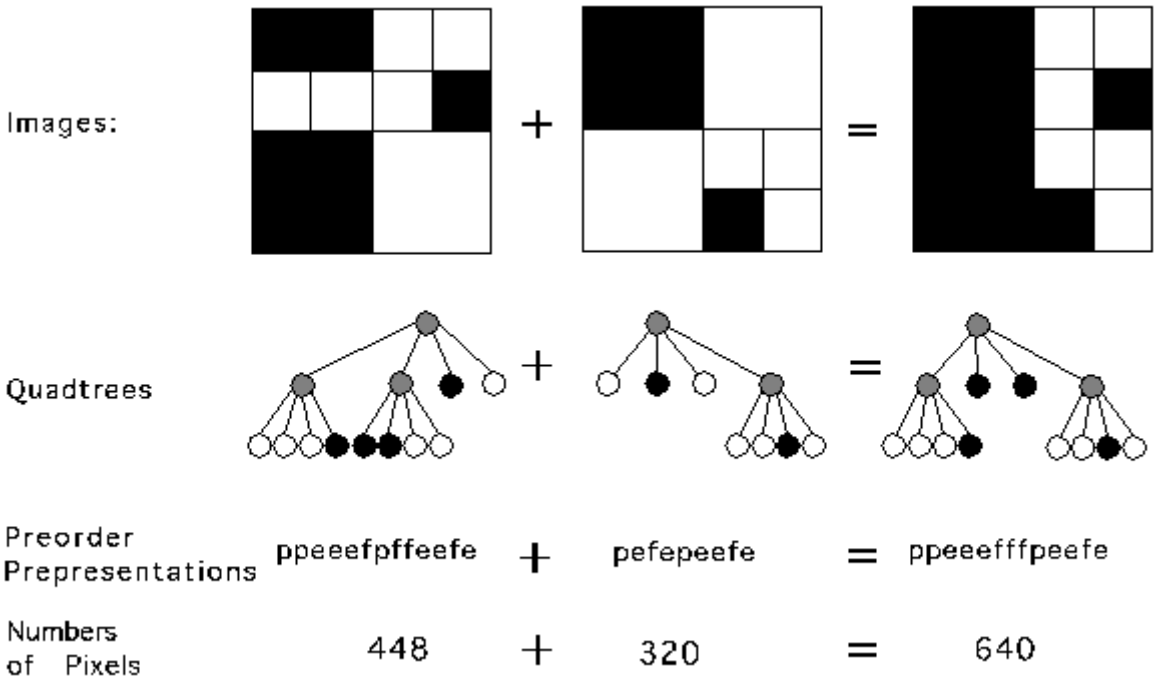


Of course, if the whole image is a single color, it can be represented by a quadtree consisting of a single node. In general, a quadrant needs only to be subdivided if it consists of pixels of different colors. As a result, the quadtree need not be of uniform depth.

The preorder representation of a quadtree consisting of a single node is given by `e', if the node represents an `empty' (white) quadrant, or `f', if the node represents a `full' (black) quadrant. The preorder representation of a quadtree of more than one node consists of the letter `p' (of "parent") followed by the preorder representation of the four subtrees in the order indicated in the figure below. See the figure for an example.

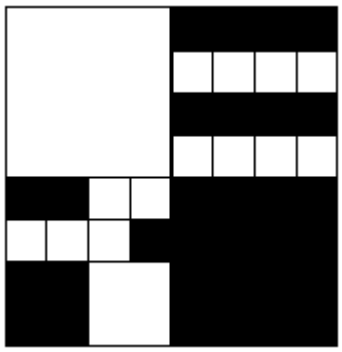
The artist believes in what is called the *preferred fullness*: for an image to be attractive the most important property is the number of filled (black) pixels in the image. Your job is to write a program that, given the quadtree representation of two images, calculates the number of pixels that are black when the two images are added.

In the figure, the first example is shown (from top to bottom) as image, quadtree, pre-order string and number of pixels. The quadrant numbering is shown at the top of the figure.



Questions

1. Give a preorder representation of the quadtree encoding the image below.



2. What is the length of a shortest and a longest string representing the preorder traversal of a quadtree encoding an image of 32 x 32 pixels? Explain your answer.

3. Write a program satisfying the specification below.

Input specification

The first line of input specifies the number of test cases (M) your program has to process. The input for each test case is two strings, each string on its own line. The string is the pre-order representation of a quadtree. It is guaranteed that each string in the input represents a valid quadtree.

Output specification

For each test case, print on one line the text `There are X black pixels.', where X is the number of black pixels in the resulting image.

Sample input

```
3
ppeeefpffeefe
pefepeefe
peeef
peefe
peeef
peepefefe
```

Sample output

```
There are 640 black pixels.
There are 512 black pixels.
There are 384 black pixels.
```