Midterm Exam

Programming Principles
(Spring 2016)

Choose and solve three problems among the following four problems via i-Campus. Each problem has its own entry in i-Campus. So, submit your solution to the entry that corresponds with the problem. Note that each problem is assigned a different score. Name your solution file in the format: "<student ID>-<problem code>.c". For example, "2015311999-A.c" or "2015311999-B.c".

A. Train Swapping (20 Points)

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 pillar 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 direction, 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 that routine.

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 ($0 \leq L \leq 50$). 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.

Example Input

```
3
3
1 3 2
```

4
4 3 2 1
2
2 1

Example Output

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

B. The Archeologists' Dilemma (30 Points)

An archeologist seeking proof of the presence of extraterrestrials in the Earth's past, stumbles upon a partially destroyed wall containing strange chains of numbers. The left-hand part of these lines of digits is always intact, but unfortunately the right-hand one is often lost by erosion of the stone. However, she notices that all the numbers with all its digits intact are powers of 2, so that the hypothesis that all of them are powers of 2 is obvious. To reinforce her belief, she selects a list of numbers on which it is apparent that the number of legible digits is strictly smaller than the number of lost ones, and asks you to find the smallest power of 2 (if any) whose first digits coincide with those of the list.

Thus you must write a program such that given an integer, it determines (if it exists) the smallest exponent E such that the first digits of 2^E coincide with the integer (remember that more than half of the digits are missing).

Input

It is a set of lines with a positive integer N not bigger than 2147483648 in each of them.

Output

For every one of these integers a line containing the smallest positive integer E such that the first digits of 2^E are precisely the digits of N, or, if there is no one, the sentence '`no power of 2`'.

Sample Input

1
2
10

Sample Output

7
8
20

C. Common Permutation (40 Points)

Given two strings a and b, print the longest string x of letters such that there is a permutation of x that is a subsequence of a and there is a permutation of x that is a subsequence of b.

Input

The input file contains several cases, each case consisting of two consecutive lines. This means that lines 1 and 2 are a test case, lines 3 and 4 are another test case, and so on. Each line contains one string of lowercase characters, with first line of a pair denoting a and the second denoting b. Each string consists of at most 1,000 characters.

Output

For each set of input, output a line containing *x*. If several *x* satisfy the criteria above, choose the first one in alphabetical order.

Sample Input

```
pretty
women
walking
down
the
street
```

Sample Output

```
e
nw
et
```

D. World Cup Noise (60 Points)

"KO-RE-A, KO-RE-A" shout 54000 happy football fans after their team has reached the semifinals of the FIFA World Cup in their home country. But although their excitement is real, the Korean people are still very organized by nature. For example, they have organized huge trumpets (that sound like blowing a ship's horn) to support their team playing on the field. The fans want to keep the level of noise constant throughout the match.

The trumpets are operated by compressed gas. However, if you blow the trumpet for 2 seconds without stopping it will break. So when the trumpet makes noise, everything is okay, but in a pause of the trumpet, the fans must chant "KO-RE-A"!

Before the match, a group of fans gathers and decides on a chanting pattern. The pattern is a sequence of 0's and 1's that is interpreted in the following way: If the pattern shows a 1, the trumpet is blown. If it shows a 0, the fans chant "KO-RE-A". To ensure that the trumpet will not break, the pattern is not allowed to have two consecutive 1's in it.

So in this problem what you will have to do is, given a positive integer n, determine the number of different chanting patterns of this length, i.e., determine the number of n-bit sequences that contain no adjacent 1's. For example, for n = 3 the answer is 5 (sequences 000, 001, 010, 100, 101 are acceptable while 011, 110, 111 are not).

Input

The first line contains the number of scenarios. For each scenario, you are given a single positive integer less than 51 on a line by itself.

Output

The output for every scenario begins with a line containing ‘Scenario #*i*:’, where *i* is the number of the scenario starting at 1. Then print a single line containing the number of n-bit sequences that have no adjacent 1’s. Terminate the output for each scenario with a blank line.

Sample Input

```
2
3
1
```

Sample Output

```
Scenario #1:
5

Scenario #2:
2
```