GLOBUSSOFT

TECHNOLOGY AHEAD OF TIME

Greetings From Globussoft

- Given below are 5 Programming questions, you have to solve any 3 out of 5 questions.

- These 5 questions you can attempt in any technology like C/C++, java, .Net, PHP

- To solve these 3 questions you've max. 3 hours.

- While Solving these questions you are not allowed to use any Search Engine like Google, Yahoo, Bing …

All the best for your test

Globussoft

QUESTION - 1

Pattern Matchers have been designed for various sorts of patterns. Mr. HKP likes to observe patterns in numbers. After completing his extensive research on the squares of numbers, he has moved on to cubes. Now he wants to know all numbers whose cube ends in 888.

Given a number **k**, help Mr. HKP find the k^{th} number (indexed from 1) whose cube ends in 888.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

Each test case consists of a single line containing a single integer **k** (1 <= **k** <= 2000000000000).

Output

For each test case, output a single integer which denotes the k^{th} number whose cube ends in 888. The result will be less than 2^{63}.

Example

```
Input:
1
1

Output:
192
```

QUESTION – 2

The magician shuffles a small pack of cards, holds it face down and performs the following procedure:

1. The top card is moved to the bottom of the pack. The new top card is dealt face up onto the table. It is the Ace of Spades.

2. Two cards are moved one at a time from the top to the bottom. The next card is dealt face up onto the table. It is the Two of Spades.

3. Three cards are moved one at a time…

4. This goes on until the nth and last card turns out to be the n of Spades.

This impressive trick works if the magician knows how to arrange the cards beforehand (and knows how to give a false shuffle). Your program has to determine the initial order of the cards for a given number of cards, 1 ≤ n ≤ 20000.

Input

On the first line of the input is a single positive integer, telling the number of test cases to follow. Each case consists of one line containing the integer n.

Output

For each test case, output a line with the correct permutation of the values 1 to n, space separated. The first number showing the top card of the pack, etc…

Example

```
Input:
2
4
5

Output:
2 1 4 3
3 1 4 5 2
```

QUESTION – 3

Mike is frantically scrambling to finish his thesis at the last minute. He needs to assemble all his research notes into vaguely coherent form in the next 3 days. Unfortunately, he notices that he had been extremely sloppy in his calculations. Whenever he needed to perform arithmetic, he just plugged it into a calculator and scribbled down as much of the answer as he felt was relevant. Whenever a repeating fraction was displayed, Mike simply reccorded the first few digits followed by "...". For instance, instead of "1/3" he might have written down "0.3333...". Unfortunately, his results require exact fractions! He doesn't have time to redo every calculation, so he needs you to write a program (and FAST!) to automatically deduce the original fractions.

To make this tenable, he assumes that the original fraction is always the simplest one that produces the given sequence of digits; by simplest, he means the the one with smallest denominator. Also, he assumes that he did not neglect to write down important digits; no digit from the repeating portion of the decimal expansion was left unrecorded (even if this repeating portion was all zeroes).

Input

There are several test cases. For each test case there is one line of input of the form "0.dddd..." where dddd is a string of 1 to 18 digits, not all zero. A line containing 0 follows the last case.

Output

For each case, output the original fraction.

Example

Input:

```
0.2...
0.20...
0.474612399...
0
```

Output:

```
2/9
1/5
1186531/2500000
```

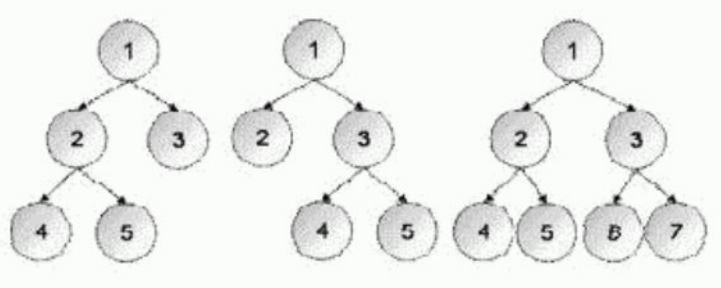
QUESTION – 4

Tree is an important data structure in Computer Science. Of all trees we work with, Binary Tree is probably the most popular one. A Binary Tree is called a **Strictly Binary Tree** if every nonleaf node in a binary tree has nonempty left and right subtrees. Let us define **a Strictly Binary Tree of depth d**, as a Strictly Binary Tree that has at least one root to leaf path of length **d**, and no root to leaf path in that tree is longer than **d**. So let us use a similar reasoning to define a generalized structure.

An n-ary Tree is called a **Strictly n-ary Tree** if every nonleaf node in an n-ary tree has n children each. A **Strictly n-ary Tree of depth d** can now be defined as a Strictly n-ary Tree that has at least one root to leaf path of length d, and no root to leaf path in that tree is longer than d.

Given the value of n and depth d, your task is to find the number of different strictly n-ary trees of depth d.

The figure below shows the 3 different strictly binary trees of depth 2.



Input

Input consists of several test cases. Each test case consists of two integers **n (0 < n <= 32), d (0 <= d <= 16)**. Input is terminated a test case where n=0 and d=0, you must not process this test case.

Output

For each test case, print three integers, n, d and the number of different strictly n-ary trees of level d, in a single line. There will be a single space in between two integers of a line. You can assume that you would not be asked about cases where you had to consider trees that may have more than 2^{10} nodes in a level of the tree. You may also find it useful to know that the answer for each test case will always fit in a 200 digit integer.

Example

```
Input:
2 0
2 1
2 2
2 3
3 5
0 0
```

```
Output:
2 0 1
2 1 1
2 2 3
2 3 21
3 5 58871587162270592645034001
```

QUESTION – 5

There are *n* dealers in the market. Each of them has some unique goods (nobody else has the same goods). Besides, each of them wants to obtain some other goods, which exist in the market. This is rather strange, but for each kind of goods on the market there exists exactly one dealer who wants to obtain it.

To prevent fraud, only exchanges in pairs are allowed in this market. Moreover, each dealer is allowed to make at most one exchange a day. But the total number of transactions isn't limited. A transaction means that all the goods of one dealer are exchanged for all the goods of the other participating dealer (partial transactions are not allowed).

You are to write a program which outputs the minimum number of days needed for each dealer to get the goods that he wants. Also output one of the possible variants of exchanges leading to this goal.

Input

The first line contains an integer *n* [*n* <= 5000]. In the second line exactly *n* numbers of goods are given, which the dealers require. If integer *j* appears as the *i*-th at input, then this means that goods required by dealer *i* are initially owned by dealer *j*.

Output

You must output the minimum number of days *m* which are needed to complete the transactions. In the next *m* lines you must output the way these transactions should be managed by the dealers. One line corresponds to one day. At the beginning of each line you must output the number of transactions on this day. After that output the pairs of dealers who exchange their goods on this day. Dealers in pairs are separated by '-' symbol. If there are many ways to perform the exchanges then output any of them.

Example

```
Input:
7
2 1 3 5 6 7 4

Output:
2
3 1-2 4-5 7-6
1 5-7
```