A. Antiplagiarism

time limit per test: 30 seconds
memory limit per test: 512 megabytes
input: input.txt
output: output.txt

The participants of this round will need to check their strength in solving an unusual problem.

At each round we unfortunately face cheating and foul play. Quite often the participants send somebody else's solutions from their accounts or simply use multiple accounts to reduce the penalty for incorrect attempts. Note that such conduct violates both the Codeforces rules and basic human ethics. When the first experience goes unpunished, it creates the illusion of 'it's okay' that motivates resort to dishonest moves again.

As you solve the problem of this round, you have a real chance to help us make Codeforces better.

We suggest to you to write your own version of the automated search of plagiarism. We understand that the condition of the problem will not be formal and the tests to your solution will not be easy to make. It often happens that the tasks from the real life aren't formalized as easily as the olympiad problems, but on the other hand, the results of your work in this case may be more useful and applicable.

Since the proposed problem is unusual not only for the participants but also for the jury, we reserve the right to make changes in the condition, scoring system, tests during the competition.

Write a program that will analyze the specified files on plagiarism and display all files in groups, grouping together the original and ones derived from it. In an ideal world we would like to see all these groups at the size of one (i.e., no plagiarism), but in practice it doesn't always happen.

We will try to prepare the test data so that the facts of plagiarism were fairly obvious to a person, and the cheating relation is transitive. We will take possible methods for solution modification from some of the fraudulent solutions of the participants of Codeforces contests and other competitions. In most tests, the files will be taken from real events, but they can also be supplemented with modified solutions that we will prepare yourselves.

We are interested in making the winner's code work as correctly as possible on real rounds so that in preparing tests we will try not to be carried away from the patterns used in practice.

You can look on the file extension to roughly determine the programming language. In a significant number of tests we will focus on languages that are popular in competitions. For example, this means that in many tests, most files will have the extension `cpp`. The list below contains the extensions used for some popular languages.

- C/C++: `cpp`,
- Java: `java`,
- Pascal/Delphi: `pas`,
- Python: `py`.

As the list of Codeforces languages is constantly updated, then your program should be expected to work for solutions in languages unknown to her. In a number of tests you will be offered solutions in rare, unpopular languages with their own extensions — your program must handle them correctly. It is acceptable (though rarely) that the couple of fraudulent statements is written in different languages.

The result of your program on each test will be evaluated by an integer number of points from 0 to 100 as a result of rounding the following expression: $a \cdot 100.0 / b$, where a is the number of pairs of fraudulent solutions found by your program, and b is the number of such pairs in the opinion of the jury. If your program indicates that some pair is fraudulent, but the jury does not agree, then your decision will receive 0 points on this test. If there are no fraudulent pairs in a test and you solution printed the same verdict, you will get 100 points.

For local testing and better understanding of the problem you are given an archive with 10 tests and answers of the jury for these tests. During the competition, you can send your solutions to be tested on 20 tests, the first 10 of which

coincide with the tests from the public archive. The remaining 10 tests are confidential, but the results of your program to them assess the quality of your solution. You can download public archive by the link http://assets.codeforces.com/files/vkcup-2015-wildcard-2-samples-1.1.zip

After the end of the contest (i.e., a week after its start) the last solution you sent will be chosen to be launched on the final tests. The final tests are confidential and distinct from those that will be used during the competition. The total number of points that will be scored in the final tests will determine the winner of the competition.

Specially for this round we implemented feature to submit ZIP-archive with multiple source files. All files should be in root of ZIP-file, no directories are allowed. You can use this feature for:

- Java 8: `main` should be in a class `Main` of default package,
- GNU C++ 11: exactly one file should contain the entry point `main`, all files to compile should have extension `cpp`.

Input

The first line of the input file `input.txt` contains integer n ($1 \le n \le 100$) — the number of files to analyze. Next n lines contain the names of n files, one name per line. All these files contain the solutions to some problem from a programming competition and are located in the current directory of your program. Trivial problems with very short solutions will not be included into tests.

Output

Print all files in groups, join in one group pairwise fraudulent solutions.

In the first line of the output file `output.txt` print t — the number of the groups you found (you need to consider only the groups that contain more than one file). Then print t lines, each line should contain two or more file names that are put in the same group by your solution. Each file should go to at most one group.

You can print both groups and file names within each group in any order.

Examples

input
6 10713587.cs 49_38a.cpp 639_536.cpp 91_a1.cpp 9l_3_2dEd.cpp t61_31.cpp

output
2 49_38a.cpp t61_31.cpp 91_a1.cpp 9l_3_2dEd.cpp

Note

This test corresponds to the first test from the public test archive.