Problem 1 – PHP Variables

Misho is a big fan of the PHP programming language, but he doesn’t like C#. One day his boss decided to start developing a new cloud-based PHP IDE with autocomplete options. He wanted from Misho to write a C# program that extracts all variable names used in a valid PHP code. The PHP IDE must be fast, so Misho only has part of the second to parse all variables from 1 MB PHP source code.

PHP is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages. The syntax of the PHP is similar to C#. As in C#, PHP requires instructions to be terminated with a semicolon at the end of each statement.

Variables in PHP are represented by a dollar sign (**$**) followed by the name of the variable. The variable name is case-sensitive. A valid variable name starts with a Latin letter or underscore, followed by any number of Latin letters, numbers, or underscores. One strange thing about PHP is that variables can be used without being initialized.

Every popular programming language has arrays. PHP also has arrays. The syntax of the arrays in the PHP is quite simple. To access the array variable you need to specify the array variable name and a key, written in a pair of square brackets ('[' and ']'). The arrays are also regular variables, so Misho needs to extract their names, too. Note that “`$arr['text']`”, “`$arr[$arrid]`”, “`$arr["text"]`” and “`$arr[0]`”are all valid uses of an array variable named “`arr`”.

As you know, a string represents a series of characters. PHP supports strings. The strings in PHP can be enclosed by single quotes “ ' ” or by double quotes “ " ”. In PHP you are allowed to use variables inside strings just by specifying their names preceded by the dollar sign (**$**). See the example bellow.

Some characters in the PHP strings must be escaped with a backslash “\” (for example the quotes used for the string enclosure, the dollar sign or the backslash itself, just like in C#). If a string is within single quotes, then you can use double quotes without escaping it and vice versa. If you escape the dollar sign in a string, PHP won’t interpret it as a variable. It will be interpreted as a regular dollar character.

PHP also supports comments. There are 2 styles of comments: “one-line comments” and “multi-line comments”. The "one-line" comment styles only comment to the end of the line (starting from the first occurrence of “//” or “#”, which is not part of a PHP string). Multi-line comments start with “/*” and end with “*/” (like C# multi-line comments, again if not part of a PHP string). All variables and strings inside comments must be ignored. Examples:

```
<?php
   # this is a lonely comment line
   /*
      $this_is_not_a_variable
      "And this is not a string"
   */
   $just = '"code"...{$valid_var}'; // This is an one-line comment
   $another = "/* This is not a comment... */";
   $Inception = "// ... neither \"this\", but this is a 'variable': $var";
?>
```

For your program to work correctly you don’t really need to know the entire PHP syntax, just pay special attention on the PHP arrays, strings and comments.

Write a program in C# to help Misho with his task.

Input

The input data should be read from the console.

The input data will be a valid PHP code from which you need to extract all variable names.

The first input line will always be equal to “<?php” and the last input line will always be equal to “?>”.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console.

In the first output line your program must print the number N of the unique variable names used in the given PHP code.

The variable names (without the dollar sign) must be printed on the next N lines sorted in ascending order. Print them character by character, using the ASCII codes of the characters.

Constraints

- The input string will always be valid PHP code starting with “<?php” and ending with “?>”.
- All variable names in the code will be valid PHP variable names and there is no need to check them explicitly.
- The input string will be no longer than 1 000 000 characters.
- The number of the unique variable names in the code will be no bigger than 10 000.
- Allowed working time for your program: 0.8 seconds. Allowed memory: 16 MB.

Examples

Input example	Output example

<?php
  $browser = $_SERVER['HTTP_USER_AGENT'] ;
  $arr = array();
  $arr[$zero] = $browser;
  var_dump($arr);
?>
	4
	_SERVER
	arr
	browser
	zero

<?php
/* This is $var1 in comments */
$var3 = "Some string \$var4 with var escaped.";
echo $var5; echo("$foo,$bar");
// Another comment with variable $var2
?>
	4
	bar
	foo
	var3
	var5

<?php # this is $comment $valid_var='"text"...{$valid_var}'; $just="Just another var $Just...";$just=$code; ?>	4 Just code just valid_var

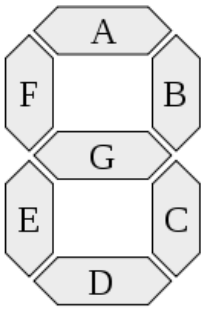
Problem 2 – 7-Segment Digits

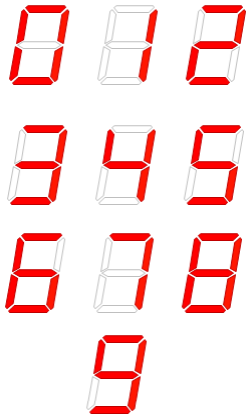
Anton acts like a natural disaster. He breaks almost everything around him. One day he broke the display of his calculator and now you must help him restore the numbers on his calculator’s display.

The calculator’s display contains **N** digits. Each digit is presented by 7 segments (A, B, C, D, E, F and G) as shown in the pictures below. Each of the segments has only 2 states (1 – on and 0 – off). When some exact segments are on, then a digit from 0 to 9 is shown (look at the table below). Unfortunately, some of the segments are not working (and of course Anton is guilty for that).

	A	B	C	D	E	F	G
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1

	A	B	C	D	E	F	G
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1





You are given the number **N** – the number of digits on Anton’s calculator display, together with the states of all 7 segments on these digits. These segments (from A to G) are given by sequences of zeros and ones.

Your task is to write a program that finds all possible combinations of digits on Anton’s broken calculator

Input

The input data should be read from the console.

From the first input line you must read the number **N** and from each of the next **N** lines you must read the states of these **N** digits’ segments, given by sequence of zeroes and ones with exact length 7.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console.

In the first output line you must write the number (**K**) of all possible combination of digits on the given calculator and in the next **K** lines you must write these combinations sorted in ascending order.

Constraints

- **N** will be between 1 and 18, inclusive. The total number of answers up to 50 000.
- Allowed working time for your program: 0.2 seconds. Allowed memory: 16 MB.

Examples

Input example	Output example
1 1011111	2 6 8

Input example	Output example
2 1111110 1111111	2 08 88

Problem 3 – Tubes

Marto is a well-known Pernik fighter. He has **N** tubes with different sizes. Marto also has **M-1** friends. His friends also need tubes for fighting.

Help Marto to cut his own tubes into **exactly M parts with same sizes**. This size should be maximal possible (bigger tube = better fighter).

Your task is to write a program that finds the biggest possible size of the **M** tubes which you can cut from the initial Marto’s tubes.

Input

The input data should be read from the console.

On the first input line your program should read the integer **N** – the number of Marto’s tubes.

On the second input line there will be the number **M** – the count of the tubes Marto needs.

On the next N lines your program should read the sizes of the Marto’s tubes Marto.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console.

On the only output line you should print the maximal possible size of the **M** tubes. If it is impossible to cut the tubes with the given criteria write “-1” on the only output line.

Constraints

- **N** will be between 1 and 20 000, inclusive.
- **M** will be between 1 and 2 000 000 000, inclusive.
- The sizes of the tubes will be between 1 and 2 000 000 000, inclusive.
- Allowed working time for your program: 0.1 seconds.
- Allowed memory: 16 MB.

Examples

Input example	Output example
3 6 100 200 300	100

Input example	Output example
4 11 803 777 444 555	200

In the first example we can cut the tubes into 6 parts (each with size of 100).

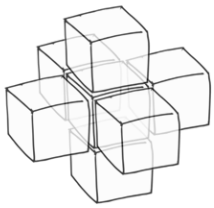
In the second example we can cut the first tube into 5 parts (200, 200, 200, 200 and 3), the second tube into 4 parts (200, 200, 200 and 177), the third tube into 3 parts (200, 200 and 44) and the fourth tube into 3 parts (200, 200 and 155). After this cuts we have exactly 11 tubes with size of 200. The cuts we made are optimal. We can’t cut the tubes into 11 parts of size 201.

Problem 4 – 3D Stars

You are given a **rectangular cuboid** of size **W** (width), **H** (height) and **D** (depth) consisting of **W** * **H** * **D** cubes, each colored in some color. Each **color** is denoted by a unique capital letter from the Latin alphabet: 'Y' is yellow, 'R' is red, 'B' is blue, 'G' green, etc. A **3D star** is a figure of size 3 x 3 x 3 consisting of 7 cubes colored in the same color staying in the following configuration: one cube is the star center and 6 cubes are stuck at its 6 sides (see the figure below).

The figure below shows a sample cuboid consisting of 7 x 4 x 3 colored cubes (width = 7, height = 4, depth = 3) and how the 3D star figure looks like:

Layer 0	Layer 1	Layer 2
B R Y Y Y Y Y	R Y Y Y Y G Y	Y R Y Y Y Y Y
Y Y Y G B G Y	Y Y Y Y G G G	Y Y Y G G G Y
R Y B Y G Y Y	R Y Y Y Y G Y	R Y B Y G B B
R Y B Y G Y Y	R B Y Y G Y Y	R Y B Y G B B



Your task is to write a program that finds how many 3D stars exist in the cuboid. At the cuboid above there are 4 stars: 3 yellow and 1 green. A cube is allowed to be shared between several stars.

Input

The input data should be read from the console. At the first line 3 integers **W**, **H** and **D** are given separated by a space. These numbers specify the width, height and depth of the cuboid. At the next **H** lines the colors of the cubes in the cuboid are given as **D** sequences of exactly **W** letters. Each sequence of **W** letters is separated from the next with a single space.

The input data will be correct and there is no need to check it explicitly.

Output

The output data should be printed on the console.

At the first line of the output print the **total number of 3D stars** in the cuboid. On the next few lines print the stars in the cuboid: **color** followed by a space and by **amount** found in the cuboid. Only colors with non-zero amount of stars should be listed. The colors should be listed in alphabetical order.

Constraints

- The numbers **W**, **H** and **D** are all integers in the range [3…150].
- The letter sequence in the input consists of capital Latin latters only
- Allowed work time for your program: 0.25 seconds.
- Allowed memory: 32 MB.

Examples

Input	Output
4 3 5 AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA	6 A 6

Input	Output
7 4 3 BRYYYYY RYYYYGY YRYYYYY YYYGBGY YYYYGGG YYYGGGY RYBYGYY RYYYYGY RYBYGBB RYBYGYY RBYYGYY RYBYGBB	4 G 1 Y 3

Problem 5 – Brackets

You are given a sequence of characters, containing only “(“, “)” and “?”.

Using this pattern write a program to count the number of possible ways, in which you can replace each of the “?” characters with a bracket character “(” or “)”, resulting in a correct bracket expression.

If you take a correct math expression and remove everything else except the brackets you will receive a correct bracket expression. For example: there are 2 correct bracket expressions with length exactly 4: `()()` and `(())`.

Input

The input data should be read from the console.

The brackets pattern will be given on the only input line.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console.

You should print the number of correct bracket combinations on the only output line.

Constraints

- The length of the character sequence (the pattern) will be between 1 and 1001, inclusive.
- Allowed working time for your program: 0.2 seconds.
- Allowed memory: 16 MB.

Examples

Input example	Output example	Explanation
`????(?`	2	The solutions are: `()()()` and `(())()`
`()(?`	1	The only possible solution is `()()`
`??????`	5	There are 5 possible solutions: `((()))`, `(())()`, `()(())`, `(()())` and `()()()`