Задача 2. The Flights

Имя входного файла:	`filghts.in`
Имя выходного файла:	`flights.out`
Ограничение по времени:	20 seconds
Ограничение по памяти:	256 Mebibytes

John and Brus recently bought several small airplanes. And now they want to make some money by holding some flights. There are N cities with airports and it's possible to hold a flight from any city to any other. Somehow Brus found out that the most efficient way is to hold N flights in such way that for each city there is exactly one incoming and one outgoing flight.

John: What is this button for?
Brus: Damn, I've forgotten my parachute!

Some cities are connected by two-way roads. There can be more than one road between two cities. Two cities are called connected if it's possible to get from one city to another by roads (possibly via some other cities). John thinks that it's too silly to hold a flight between two connected cities. Thus he wants to assign all N flights in such way that there will be no flights between connected cities. You have to calculate the number of flight assignments that satisfy John's condition modulo 1234567891.

Формат входного файла

The first line contains single integer T ($1 \le T \le 47$) — the number of test cases. Each test case starts with a line containing two integers N and M — the number of cities and the number of roads respectively ($1 \le N \le 1000,\ 1 \le M \le 10^5$). Each of next M lines contains two different integers A_i and B_i — the cities connected by i-th road ($1 \le A_i, B_i \le N$). All the integers in a single line are separated by single spaces.

Формат выходного файла

For each test case print a single line containing the number of flight assignments satisfying John's condition modulo 1234567891.

Пример

filghts.in	flights.out
2 4 3 1 2 3 4 4 3 10 0	4 1334961

Задача 3. Develop a Water-Pipe

Имя входного файла:	`develop.in`
Имя выходного файла:	`develop.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Recently you spent a lot of time playing the following game: there is a water-pipe, which is located on 2-dimensional plane and consisted of square tiles.

Each tile is either `I`-tile, which connects left and right edges of tile or top and bottom ones, `L`-tile, which connects left and top, or right and top, or left and bottom, or right and bottom edges of tile, `T`-tile, which connects any three edges, or, finally, `A`-tile, which connects all four edges.

Also there's a *sink* and *source* tiles. *Sink* tile connects its right edge with some external water-tank, while *source* node connects its left edge with the bath. There may be more that one *sink* and *source* nodes.

Water-pipe is *well-formed*, if all the tiles are placed in such a way, that if some tile connects, for example, its left edge with anything else, then tile, located on the left of that tile, necessarily connects its right edge with some other edges (or water-tank, or bath).

Well-formed water-pipe doesn't require *sink* to be connected to *source*.

Before the game started, all the tiles are rotated some times clockwise for 90°, and then your aim is, rotating them, restore *well-formed* water-pipe.

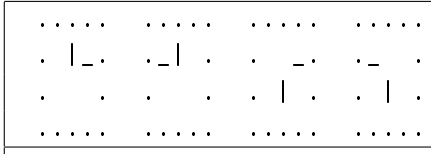
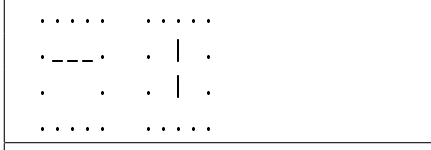
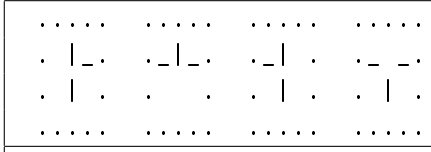
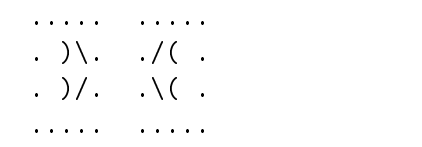
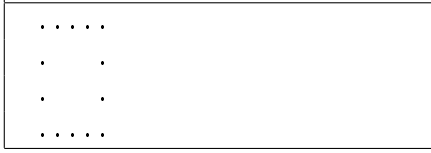
After some easy levels rules of the game got a little bit more difficult. Especially, the water-pipe can't be made well-formed until you change some `T`-tiles to `A`-tiles.

After some time you have shown this game to your friend, and he began to solve levels much faster then you did.

You don't want to be beaten by your friend and now you need to write a program, which will solve this game in order to solve all remaining levels before you friend has done it.

Формат входного файла

First line of input contains two integer numbers — N and M — dimensions of field. Then $3N+1$ lines follow, each contains $4M+1$ characters. Each tile is defined by 4 rows of 5 characters according to the following table (only dot, underscore, space and vertical slash symbols are used):

	L-tiles
	I-tiles
	T-tiles
	A-tiles
 Note: here dot, slash, backslash, space and parentheses symbols are used	*Sink* and *Source* respectively
	Empty tile

$1 \le N \le 25$
$1 \le M \le 25$
For more details see sample input.

Формат выходного файла

Output *well-formed* water-pipe, obtained from given one by rotating any number of tiles and by replacing some (possibly zero) `T`-tiles to `A`-tiles in the same format, as in input file. If it is impossible to obtain *well-formed* water-pipe, output file must contain single word `IMPOSSIBLE` without quotes

Пример

develop.in	develop.out
3 6 ..... .)\.___. | ._| . . . .)/. . | . . . . .___. . . . . ._| .___./(. . . . . . .\(.	..... .)\.___.___._ . . . .)/. . . | . . . . | . . . . | . . . . |_.___./(. . . . . . .\(. .....
3 6 ..... .)\._ _. | ._| . |_./(. .)/. | . | . | . |.\(. ..... . |_._|_._| .___. | . . . . | . . .. |_._|_._|_._|_._ _./(. . . . . . | .\(.	..... .)\._ _._____._ _._ _./(. .)/. | . . | . | .\(. ..... . _._|_._ . | . | . . . | . | . | . | . | . . .. |_._|_._|_._|_._|_._|_./(. . . . . . .\(. .....
3 6 ..... .)\.___. | ._| . |_./(. .)/. . | . | . | .\(. ..... .)\._|_. .___. | . . .)/. . . . | . . .. ._| ._ _./(. . . . . . | .\(.	IMPOSSIBLE

Задача 5. JavaScript Interpreter

Имя входного файла:	`js.in`
Имя выходного файла:	`js.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

In this problem you have to implement simplified interpreter of JavaScript language, which operates on simplified document object model of an HTML page.

Page is going to be shown in the text browser with resolution of 20 rows and 40 columns.

Initially, before executing JavaScript code, page is absolutely empty, and document model contains the only element — *document*.

To create new element method *createElement* of *document* object is used. It takes one parameter, which for this problem is always equal to `"span"` in double quotes, and returns the pointer to the created element.

Each element has following attributes:

attribute name	**constrains**	**description**	**default value**
left	integer $-100 \le left \le 100$	Offset from parent element's left border	0
top	integer $-100 \le top \le 100$	Offset from parent element's top border	0
width	integer $0 \le width \le 100$	Width of the element	10
height	integer $0 \le height \le 100$	Height of the element	10
border	boolean *true* or *false*	Whether element has border or not	*false*
zIndex	integer $0 \le zIndex \le 100$	Elements with higher zIndex are drawn over the ones with lower zIndex	0
innerText	string From 0 to 100 characters	Text inside the element	*Empty string*
align	string «center» or «left» or «right»	Text align inside the element	«left»

Program may contain any number of variables, where variable name is a set of English letters, digits and underscore characters. First character of variable name is always letter. Variable names are case sensitive.

Each variable has to be defined before it is first time used. Initially the only defined variable is *document*. There are four types of variables in this problem: element, integer, string and boolean.

To define new variable the following constructions are used:

`var` *variable_name* = *value*;

or

`var` *variable_name* = *other_variable_name*;

Where *value* is either integer number, string in double quotes, true or false or calling of *createElement* method. It uniquely defines type of the variable. Note, that after command

```
var moo = "5";
```

`moo` have type string, not integer.

To change value of some variable following constructions are used:

variable_name `=` *value*;

or

variable_name `=` *other_variable_name*;

All elements are stored as references. It means that executing following code:

```
var moo = document.createElement( "span" );
var q = moo;
q.width=100;
```

Will change width of both `q` and `moo` (because `q` and `moo` point to the same element).

Though, the following code

```
var moo = document.createElement( "span" );
var q = moo;
q = document.createElement( "span" );
q.width=100;
```

Will not change width of `moo`, because `q` points now to other element.

All strings, booleans and integers are stored by value, not by reference.

To change any attribute except innerText of some element the following construction is used:

element`.style.`*attribute_name* `=` *value*`;`

or

element`.style.`*attribute_name* `=` *variable_name*`;`

To change *innerText* the same construction, but without *style* member, is used.

Members of *style* and *innerText* are also variables, so the following code is OK:

```
moo.style.left = q.style.top;
```

To add child element to an existing one following construction is used:

element1`.appendChild(` *element2* `);`

Where *element2* is either variable, which contains element, or calling of *createElement* method, and *element1* is either already created span element or *document*. *element1* doesn't point to any descendant of *element2* or to *element2* itself.

You may assume, that code consists only of described constructions, as well as all operations of changing attribute use values or variables of appropriate type (for example there will never be construction which tries to assign string value to *left* attribute).

document has type *element* (slightly extended by adding method *createElement*), but the only properties of it which may be changed are *innerText* and *align*.

Commands are separated by semicolon character. Any number of whitespace characters may be used anywhere but inside variable names, methods, variable values or attribute names. Whitespaces are characters with ASCII codes 9, 10, 13 and 32 ('`\t`', '`\n`', '`\r`' and ' ' respectively).

Your aim is to execute program and render the page after performing all operations.

The following rules must be followed during rendering page:

1. If two elements (which have the same parent) overlap, the one with higher zIndex is rendered on the overlapped area. If their zIndex-es are equal, the one which was added to the parent element later is rendered on the overlapped area.

2. If element doesn't have border, it changes to space every character of the parent with coordinates x, y for which both *element's left* $\leq x <$ *element's left* + *element's width* and *element's top* $\leq y <$ *element's top* + *element's height*.

3. If elements does have border, it additionally surrounded by `'+'`, `'-'` and `'|'` characters (see sample output for details). Border is drawn outside the area filled by the span.

4. Text is written inside the span. If text length is higher that *width* of span, it has to be written on several lines. To separate text follow the following algorithm: if after writing new word current line will not exceed *width* of the span, write it. Otherwise start new line. If word's length is higher then *width* of the span, write *width* characters on one line and repeat process for remaining part of the word starting from the next line. If text *align* is «center» and one of the lines can't be centered exactly (i.e. its length doesn't equal to *width* of the span modulo 2), one trailing space must be added to this line. If number of lines of text is more than *height* of span, all extra lines must be

omitted. Note that in this problem *innerText* is always single space separated set of English words and numbers.

5. Everything, that doesn't fit into the parent span (or screen, if parent of the span is document), including borders, must be omitted.

Формат входного файла

Input file contains code to be executed
Code won't contain more than 1000 semicolons.

Формат выходного файла

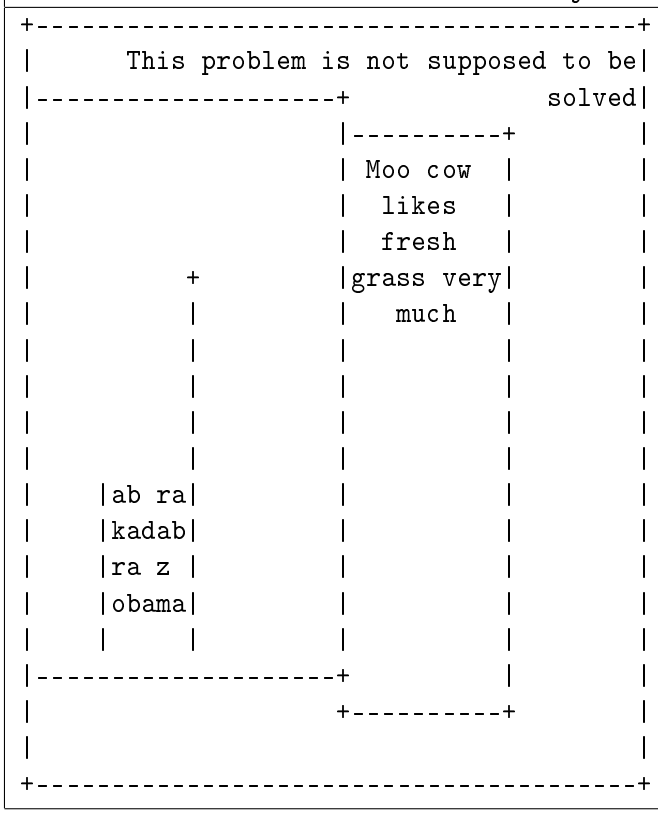
Write 22 lines, each containing 42 characters — the final state of the screen, surrounded with a border. See sample output for details.

Пример

js.in

```
var el = document.createElement( "span" );
var el2 = document.createElement( "span" );
var el3 = document.createElement( "span" );
var q = true; el.style.border = q; el2.style.border = q;
el.style.width = 20; el.style.height = 15; el.style.top = 2;
el2.style.left = 5; el2.style.top = el2.style.left;
el2.style.width = 5;
el2.innerText = "abra abra abra abra abra ab ra kadabra z obama";
el3.style.left = 21; el3.style.top = 3; el3.style.height = 15;
el3.innerText = "Moo cow likes fresh grass very much";
el3.style.align = "center";
el3.style.border = true;
el.appendChild( el2 ); el.appendChild( document.createElement( "span" ) );
document.appendChild( el );
document.appendChild( el3 );
document.innerText = "This problem is not supposed to be solved";
document.style.align = "right";
el.style.zIndex = 1;
```

js.out



Задача 7. Shortest Path

Имя входного файла:	`shortest.in`
Имя выходного файла:	`shortest.out`
Ограничение по времени:	4 seconds
Ограничение по памяти:	256 Mebibytes

Given a convex polyhedron with N faces and two points on its surface, find the shortest path between these points that goes over the polyhedral surface.

Формат входного файла

The first line of the input file contains a positive integer d ($1 \le d \le 150$) — the number of test samples.

The first line of each test sample contains an integer N — the number of faces of the polyhedron ($4 \le N \le 15$). The faces are described in the following N lines, a line for each face. The description starts with an integer K — the number of vertices of the face ($3 \le K \le N - 1$). Then follow triples of the coordinates x, y, z of all K vertices of the face. It is guaranteed that no pair of faces lie in the same plane.

The last line of each test case contains six numbers — x_a, y_a, z_a — the coordinates of the initial point and x_b, y_b, z_b — the coordinates of the terminal point.

All the coordinates in the input file are integers not greater than 1000 in absolute value.

Формат выходного файла

For each test sample the output file should contain the legnth of the shortest path between the given points over the polyhedral surface calculated with an accuracy of 2 decimal digits.

Пример

shortest.in	shortest.out
1 6 4 0 0 0 0 0 1 0 1 1 0 1 0 4 1 0 0 1 0 1 1 1 1 1 1 0 4 0 0 0 0 1 0 1 1 0 1 0 0 4 0 0 1 0 1 1 1 1 1 1 0 1 4 0 0 0 0 0 1 1 0 1 1 0 0 4 0 1 0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 1	2.24

Задача 11. Puzzle

Имя входного файла:	`puzzle.in`
Имя выходного файла:	`puzzle.out`
Ограничение по времени:	15 seconds
Ограничение по памяти:	256 Mebibytes

There is a well-known riddle called «puzzle», which has one rectangle divided into $m \times n$ square blocks. Each side of the block can be either smooth or may have either a hollow or a flange. Two blocks may adjoin their sides only when the side of the first block has a hollow, and the side of the second block has a flange. While assembling the puzzle you can revolve the block on it's center horizontally, but you cannot turn it up. Your task is to find for a given set of blocks amount of different ways to assemble the puzzle. Two assemblies are considered identical, if each of mn positions has identically oriented blocks with identical configuration. Note that two assemblies that can be transformed to each other by rotation are still considered different.

Формат входного файла

The first line of input contains a single natural number d ($1 \leq d \leq 100$) - amount of test samples.

Each test sample starts with a line containing m and n - puzzle dimensions ($1 \leq n \leq 6$, $1 \leq m \leq 5$). m lines follow with block descriptions. Each block description consists of 4 numbers, that define the side type in order of sides bypass: 0 if the side smooth, 1 if there is a hollow, and 2 if there is a flange.

Формат выходного файла

Output for each sample in a separate line a single number - amount of different ways to assemble the puzzle.

Пример

*

puzzle.in	puzzle.out
2 3 3 0 0 1 2 0 0 1 2 0 0 1 2 0 0 1 2 0 1 1 2 0 1 1 2 0 1 1 2 0 1 1 2 2 2 2 2 3 3 0 0 1 2 0 0 1 2 0 0 1 2 0 0 1 2 0 1 1 2 0 1 1 2 0 1 1 2 0 1 2 2 2 2 2 2	1 0

Задача 13. Pisces

Имя входного файла:	`pisces.in`
Имя выходного файла:	`pisces.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

It's not easy to find a black fish in a black room.
Especially if a black cat came there first.

`# man cat`

Helge works for the RIEL (Research Institute of Extraterrestrial Life-forms). Her latest mission to the Pisces constellation was successful: n wonderful, active, colorful specimens of fish were obtained. There's a problem, however: the fishes of the same color are absolutely indistinguishable.

Now Helge wants to mark them with colored rings. She repeats the following steps until each pair of specimens becomes distinguishable:

- select some color that has not yet been used (not even as the natural color of some fish),
- get k random specimens from the aquarium (each subset of cardinality k is equally likely to be chosen),
- mark each of them with rings of selected color.

Each procedure (three steps) takes exactly one minute.

Two specimens are considered to be distinguishable if their colors differ or sets of colors of their rings differ.

The only thing Helge is unsure about now is the expected time before her task will be complete. So she asked you to write a program to compute this value.

Формат входного файла

Input consists of two lines. First line contains two integers n and k ($1 \le k < n \le 30$). Second line contains n integers, i-th integer describes the initial color of i-th specimen. The integers which describe colors are positive and don't exceed 1000.

Формат выходного файла

Write one real number: the expected time (in minutes) before each pair of specimens becomes distinguishable. Your answer will be considered correct if the relative or absolute error is within 10^{-6}.

Примеры

pisces.in	pisces.out
2 1 1 1	1.000000
2 1 30 239	0.000000

Задача 17. Server

Имя входного файла:	`server.in`
Имя выходного файла:	`server.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Peter likes communication using his PC very much. He has recently moved to a new project. He learned that his friends who live in the neighborhood have computers too, and he proposed to join their PC's into a network. The friends agreed. They pulled cables from Peter's computer to their own. In a while the friends' friends connected to their PC's. Each new PC was connected only to a PC which was already in the network. No two PC's connected to the network had to be linked by wires additionally. This way, a network of N computers appeared. The friends were exchanging information with each other but finally understood that they needed more servers. They decided to turn some of their PC's into proxy servers. The computer community of the project can install K servers, no more, no less. The task now is to decide which PC's are to be turned into proxy servers. The main criterion is the monthly toll for servers serving other PC's.

Each PC is assigned a service fee. The cost of serving one PC by a server equals the fee multiplied by the length of the wire connecting this PC to the server which serves this particular PC.

Your task is to write a program which will distribute the K servers in such a way that the common expenses on servers are minimal.

Формат входного файла

The first line of the input file contains two integers N and K. N is the number of PC's in the network and K is the number of servers to be installed ($1 \le N \le 300,\ 1 \le K \le 20$). All PC's in the network are numbered from 1 to N. Peter's PC is number 1. The second line contains an integer - the service fee for the first computer.

The next $N-1$ lines each contain three integers separated by spaces — C_i, L_i, T_i — the information about other PC's in the network according to their number. C_i is the number of the PC through which the i-th PC was initially connected to the network, L_i is the length of the cable connecting these PC's, and T_i is the service fee for this particular PC ($2 \le i \le N,\ 1 \le C_i \le N,\ 1 \le L_i, T_i \le 10^4$).

Формат выходного файла

The first line of the output file must contain one integer — the minimal service cost of all PC's by all servers. The second line must contain the numbers of PC's which are to be turned into proxy servers, separated by spaces.

Пример

server.in	server.out
3 1 2 1 2 10 1 3 3	19 2
3 2 2 1 2 10 1 3 3	4 2 3

Задача 19. K-equivalence

Имя входного файла:	`kequiv.in`
Имя выходного файла:	`kequiv.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Consider a set K of positive integers.

Let p and q be two non-zero decimal digits. Call them K-equivalent if the following condition applies:

> For every $n \in K$, if you replace one digit p with q or one digit q with p in the decimal notation of n then the resulting number will be an element of K.

For example, when K is the set of integers divisible by 3, the digits 1, 4, and 7 are K-equivalent. Indeed, replacing a 1 with a 4 in the decimal notation of a number never changes its divisibility by 3.

It can be seen that K-equivalence is an equivalence relation (it is reflexive, symmetric and transitive).

You are given a finite set K in form of a union of disjoint finite intervals of positive integers.

Your task is to find the equivalence classes of digits 1 to 9.

Формат входного файла

The first line contains n, the number of intervals composing the set K ($1 \le n \le 10\,000$).

Each of the next n lines contains two positive integers a_i and b_i that describe the interval $[a_i, b_i]$ (i. e. the set of positive integers between a_i and b_i, inclusive), where $1 \le a_i \le b_i \le 10^{18}$. Also, for $i \in [2..n]$: $a_i \ge b_{i-1} + 2$.

Формат выходного файла

Represent each equivalence class as a concatenation of its elements, in ascending order.

Output all the equivalence classes of digits 1 to 9, one at a line, sorted lexicographically.

Пример

kequiv.in	kequiv.out
1 1 566	1234 5 6 789
1 30 75	12 345 6 7 89