Problem A. Boxes

Input:	*standard input*
Output:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are given n three-dimensional boxes. For each box you know its size in each of the dimensions. You are allowed to rotate and flip them thus choosing which of the dimensions will be length, which will be width and which will be height.

You are allowed to put one box on top of the other if the length of the top box is no greater than the length of the bottom one and the width of the top box is no greater than width of the bottom one. In other words, the box should fit on the box we want to put it on. Your goal is to build the tower of maximum possible height.

Input

The first line contains one integer n ($1 \le n \le 30$). Each of the next n lines contain three integers a_i, b_i and c_i ($1 \le a_i, b_i, c_i \le 10^6$) — dimensions of the i-th box.

Output

Examples

standard input	standard output
3 3 1 3 2 2 2 1 2 1	5 2 1 1 3 3 3 1 1 2

Note

First print the maximum possible height H. Then print integer k — the number of boxes in the best possible tower. Each of the next k lines should contain four integers describing a box — its index m_i, length l_i, width d_i and height h_i. Print boxes from bottom to top of the tower. Boxes are numbered from 1 to n in order they appear in the input.

Problem B. Computer

Input:	*standard input*
Output:	*standard output*
Time limit:	1 secomd
Memory limit:	256 mebibytes

Young programmer Vasya develops a large humanoid robot to help him with his homework. This robot has a computer for which Vasya is writing a control program.

To understand, how to write computer programs, Vasya wants to solve the following problem.

Initially, the computer contains only one number x in its memory. Vasya wants to get the numbers: $a_1x, a_2x, \ldots, a_nx$ in its memory, using a minimal possible number of instructions.

Now, the computer can perform only the following instructions:

1. Addition of two numbers
2. Subtraction of two numbers
3. Bitwise left shift (shift for k bits is equivalent to multiplication by 2^k)

All intermediate values are stored in memory and could be used for calculating other values. Any value that hasn't been stored in memory, couldn't be used. Before the first instruction only value of x is stored in memory.

The computer program should satisfy two constraints.

1. It shouldn't calculate values larger than $42x$. It guarantees that the computer doesn't have any overflows.
2. It shouldn't calculate negative values. So it's forbidden to subtract greater numbers from smaller ones.

The order in which numbers $a_1x, a_2x, \ldots, a_nx$ appear in memory doesn't matter.

Input

The first line contains an integer number n — the number of required numbers to store in memory ($1 \le n \le 41$). The second line contains n integer numbers a_i ($2 \le a_i \le 42$). All a_i are distinct. The number of x is not given, so your program should be correct for every x.

Output

In the first line, output a single number m — minimal possible number of instructions required. Next m lines should contain instructions descriptions in the following format.

1. Left shift of ax by k bits: "`a<<k`"
2. Addition of ax and bx: "`a+b`"
3. Subtraction of ax from bx: "`b-a`"

Instrction description shouldn't contain spaces.

Examples

standard input	standard output
3 3 5 18	5 1<<1 1<<4 1+2 2+3 2+16
1 29	4 1<<1 1<<5 1+2 32-3
4 12 19 41 42	8 1<<1 1<<4 1+2 3<<2 3+16 19<<1 3+38 1+41

Problem C. Conway

Input:	*standard input*
Output:	*standard output*
Time limit:	4 seconds
Memory limit:	512 megabytes

There are M light bulbs in a room and N switches outside of it. For the purpose of this problem, N is guaranteed to be **odd**. Each bulb has its own power p_i and can only be fully on or fully off. If the bulb is on, it adds p_i units to the illumination of the room.

Some switches and bulbs are connected by cords. Toggling a single switch changes the state of all bulbs it is connected to. There are no restrictions on connections, which means that any switch can be connected to any subset of bulbs, and vice versa.

John invented a new game and invited his friends Roland and Patrick to play it. Roland likes light and tries to make the illumination as high as possible, and the Patrick's goal is strictly opposite: to reduce the illumination as low as possible. There is a value K chosen by John to determine the winner. At the end of the game, if the total power of all bulbs turned on is greater than or equal to K, then Roland is considered the winner, otherwise, the winner is Patrick. The game process looks as follows:

- Switches are numerated from 1 to N.
- Each player makes exactly $\frac{N-1}{2}$ moves, one after another. Roland makes his move first.
- When making his i-th move, Roland can toggle switches numbered $2 \cdot i - 1$ and $2 \cdot i$. These are first and second switches on his first move, third and fourth on the second move, and so on. Note that he can choose to toggle any subset of these two switches (one of them, both or none). If any bulb is connected to both switches, it changes its state once for each toggled switch.
- Rules for Patrick are absolutely the same, the only difference is in the indices of switches he can toggle. On his i-th move, he can toggle switches $2 \cdot i$ and $2 \cdot i + 1$. These are second and third for his first move, fourth and fifth for the second move, and so on.

John likes to watch his friends playing while he already knows the result. He asks you to write a program that will determine the winner for each of the T games, assuming that both Roland and Patrick play optimally.

Input

The first line of the input contains single integer T — number of test cases ($1 \le T \le 5$). Each test case describes a separate game.

The first line of each test case contains three integers N, M and K ($3 \le N \le 33$, N is odd, $1 \le M \le 32$, $0 \le K \le 2 \cdot 10^9$) — the number of switches, the number of bulbs and illumination value to determine the winner.

The second line of each test case contains M integers l_i ($1 \le l_i \le 5 \cdot 10^7$) — the power of each bulb.

The next N lines of each test case describe connections between switches and bulbs. Each of them contains M characters. If i-th switch is connected to j-th bulb, then j-th character of i-th line equals '1', otherwise it equals '0'.

Output

For each test case, print a line with the name of the winner, that is, either "Roland" or "Patrick".

Examples

standard input	standard output
2 3 2 10 10 10 01 00 11 3 5 1 1 2 3 4 5 01011 11000 10011	Roland Patrick

Note

In the first game, one of the optimal strategies for Roland is the following one: on his first turn, he toggles the first and the second switches. For any possible Patrick's move, there will be exactly one bulb that is turned on at the end of the game, so Roland wins anyway.

In the second game, regardless of the Ronald's move, Patrick can turn off all the bulbs.

Problem D. Treasures

Input:	*standard input*
Output:	*standard output*
Time limit:	1 second
Memory limit:	256 megabytes

Daughter of the king of Flatland plans to marry charming prince. This prince wants to present her some treasures, but he is not sure which gems to pick from the set he has.

Prince has n gems, the i-th of them has weight w_i and costs v_i. Prince wants to pick the most expensive subset with the total weight of at least L. However, the princess won't accept the gift that weights more than R.

Your goal is to find the subset of maximum possible total cost and weight from L to R inclusive.

Input

The first line contains integers n, L and R ($1 \le n \le 32$, $0 \le L \le R \le 10^{18}$). Next n lines describe gems. Each of these lines contains two integers w_i and v_i ($1 \le w_i, v_i \le 10^{15}$) — weight and cost of corresponding gem.

Output

The first line of the output should contain integer k — the number of gems in the gift. The second line should contain indices of gems forming the best possible gift.

Gems are numbered from 1 to n in order they appear in the input. If there is no way to make any gift print 0 in the only line of the output.

Examples

standard input	standard output
3 6 8 3 10 7 3 8 2	1 2

Problem E. Tower of Hanoi

Input:	*standard input*
Output:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

You are given three rods. First rod contains n disks of different size arranged in order of descending radius from bottom to top. Two other rods are initially empty. Your goal is to relocate all disks from rod one to rod two using minimum possible number of moves.

One move is to take topmost disk of some rod and put it on the top of some other rod. It is not allowed to place disk on top of some smaller disk.

Input

First line contain a single integer n ($1 \le n \le 19$) — the number of disks on the first rod.

Output

Each line of the output should describe one move by two indices — the index of rod to take topmost disk from and the index of rod to put disk. The number of moves made should be minimum possible.

Examples

standard input	standard output
3	1 2 1 3 2 3 1 2 3 1 3 2 1 2

Problem F. Game With Keyboard

Input:	*standard input*
Output:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

Masha and Misha are playing an interesting game. On the first stage each one of them writes out a set of words, and first stage of the game ends. In the second stage of the game participants take the keyboard and one by one remove the keys out of it corresponding to Latin letters, a total of l keys. The first player removes one key, and then another player removes one of the remaining keys etc.

After that, players count the number of their words that can be typed using the remaining keys. The one with a less remaining number of words loses to the opponent a number of candies equal to the difference in the numbers of remaining words.

The first stage of the game has already been completed, then Masha goes first. Determine how many candies she can guarantee herself with the optimal game of both.

Input

First line of input contains an integer number l — the number of keys, that will be removed by the end of the game ($1 \le l \le 26$). Then follows sets of words of Masha and Misha, each with the following format: First line contains the number of words (not greater than 15), next line contains nonempty words itself (containing not greater than 30 letters each), divided by spaces.

All words are nonempty and consist of lowercase Latin letters.

Output

Output one integer number — the number of candies Masha could win with the optimal game. If Masha loses, output her minimal loss with a minus sign.

Example

standard input	standard output
2 5 abacaba a zxxzyz trava abc 1 a	1

Problem G. Count Perfect Matchings

Input:	*standard input*
Output:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

Matching in a graph is called *perfect* if it covers all vertices of the graph.

You are given an undirected graph (possibly, not bipartite). Count the number of perfect matchings modulo 10^9+7.

Input

The first line contains two integers n and m ($1 \le n \le 30$, $0 \le m \le \frac{n(n-1)}{2}$) — the number of vertices and the number of edges in the graph.

Output

Print one integer — the number of perfect matching in the given graph modulo $10^9 + 7$.

Examples

standard input	standard output
4 4 1 3 1 4 2 3 2 4	2

Problem H. Regular Bracket Sequences

Input:	*standard input*
Output:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Consider sequences of parentheses ('(' and ')') and square brackets ('[' and ']'). Sequence s is called regular if at least one of the following is true:

- s is empty;
- t is regular, and $s =$ (t) or $s =$ [t];
- t_1 and t_2 are regular, and $s = t_1 t_2$.

You are given integer n. Print all regulard sequences of length n in lexicographical order. Brackets are compared as follows: '(' < '[' < ')' < ']'.

Input

The input consists of one integer n ($0 \le n \le 16$).

Output

Print all sequences in lexicographical order. Print each sequence on a new line.

Examples

standard input	standard output
4	(()) ([]) ()() ()[] [()] [[]] []() [][]

Problem I. Polynomial Hash

Input:	*standard input*
Output:	*standard output*
Time limit:	5 seconds
Memory limit:	128 mebibytes

Polynomial hash of string s is calculated as follows.

$$h(s) = \sum_{i=1}^{|s|} s_i \cdot p^{|s|-i} \mod (m),$$

where m is some prime integer.

You are given string s, integers m and p and have to find string $s' \neq s$ such that $h(s') = h(s)$.

Input

The first line contains string s that consists of lowercase and uppercase English letters, digits, and underscore character '_'. Length of this string doesn't exceed 14. Next line contains two integers m ($2 \leq m < 10^{12}$) and p ($2 \leq p \leq 2000$), m is guaranteed to be prime.

It is guaranteed that the answer exists, in particular $p^{13} > m$.

Output

Print string s' that has the same hash value as s. Its length should not exceed 14 characters and it is allowed to contain only lowercase and uppercase English letters, digits, and underscore character.

Example

standard input	standard output
abacaba 17239 257	J2jLwXRJCM

Problem J. Limak and Shooting Points

Input:	*standard input*
Output:	*standard output*
Time limit:	2 seconds
Memory limit:	256 megabytes

Bearland is a dangerous place. Limak cant travel on foot. Instead, he has k magic teleportation stones. Each stone can be used **at most once**. The i-th stone allows to teleport to a point (ax_i, ay_i). Limak can use stones **in any order**.

There are n monsters in Bearland. The i-th of them stands at (mx_i, my_i).

The given $k + n$ points are pairwise distinct.

After each teleportation, Limak can shoot an arrow in some direction. An arrow will hit the first monster in the chosen direction. Then, both an arrow and a monster disappear. Its dangerous to stay in one place for long, so Limak can shoot only one arrow from one place.

A monster should be afraid if its possible that Limak will hit it. How many monsters should be afraid of Limak?

Input

The first line of the input contains two integers k and n ($1 \le k \le 7$, $1 \le n \le 1000$) — the number of stones and the number of monsters.

The i-th of following k lines contains two integers ax_i and ay_i ($-10^9 \le ax_i, ay_i \le 10^9$) — coordinates to which Limak can teleport using the i-th stone.

The i-th of last n lines contains two integers mx_i and my_i ($-10^9 \le mx_i, my_i \le 10^9$) — coordinates of the i-th monster.

The given $k + n$ points are pairwise distinct.

Output

Print the number of monsters which should be afraid of Limak.

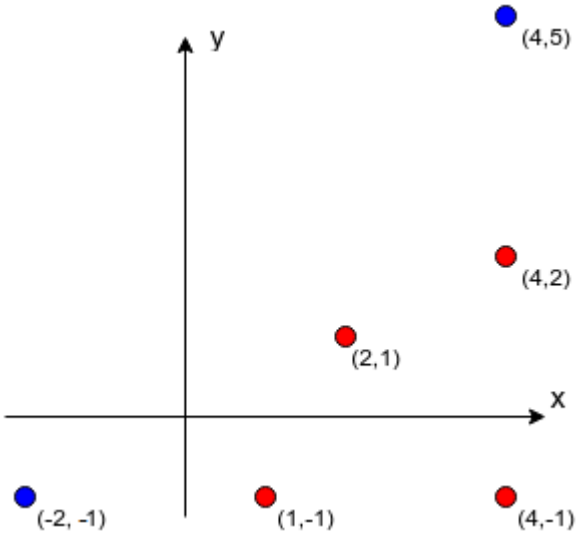
Examples

standard input	standard output
2 4 -2 -1 4 5 4 2 2 1 4 -1 1 -1	3
3 8 10 20 0 0 20 40 300 600 30 60 170 340 50 100 28 56 90 180 -4 -8 -1 -2	5

Note

In the first sample, there are two stones and four monsters. Stones allow to teleport to points $(-2, -1)$ and $(4, 5)$, marked blue in the drawing below. Monsters are at $(4, 2)$, $(2, 1)$, $(4, -1)$ and $(1, -1)$, marked red. A monster at

$(4, -1)$ shouldn't be afraid because it's impossible that Limak will hit it with an arrow. Other three monsters can be hit and thus the answer is 3.



In the second sample, five monsters should be afraid. Safe monsters are those at $(300, 600)$, $(170, 340)$ and $(90, 180)$.