Problem A. Balanced Number

Input file:	`balanced.in`
Output file:	`balanced.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

Let us say a number is *balanced* in k-based numeric system if it has some suffix that equals to some prefix in its k-base representation. For example, when $k = 12$, 1971 and 19719 are balanced, but 1415 is not. Your task is to find any number in k-based numeric system such that if we append any digit to this number, we get a balanced number.

Input

Input file contains only one number — an integer k ($2 \le k \le 24$).

Output

Output file should contain one integer number in k-based numeric system, satisfying the problem requirements. Use characters '0' ... '9', 'A' ... 'N' for its representation. This number should contain at most 2^{25} digits.

Example

balanced.in	balanced.out
2	10101

Problem B. Cabbage Problem

Input file:	`cabbage.in`
Output file:	`cabbage.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

It is well known that the favorite dish of senators in ancient Rome was meat with cabbage. The laws were passed during the meal breaks. When peace and order had come to Rome, the choice of cabbage sort became of vital importance. The problem was in the abundance of cabbage sorts while each senator had his own only favorite. If was of great necessity to have this problem solved before dinner. And as the democracy prevailed in Rome, the only way to decide the question was the vote.

According to ancient Roman traditions, the order of the vote is as follows. The first sort of cabbage is proposed. If the majority (more than a half of all the senators) vote for it then this sort is chosen as the basic usual sort and it will be served with meat. Otherwise the next sort is put to the vote etc. If none is taken then the vote is supposed to have failed and they give up eating cabbage at all.

Each senator knows the preferences of others. For example, preferences of senator A may be as follows:

#	Senator A
1	Sort 1
2	Nothing
3	Sort 2
4	Sort 3

Here senator A wants his favorite, first sort of cabbage to be accepted. If sort 1 is not accepted, then he would rather prefer none of the rest because they cause indigestion. If one of them is to be accepted anyway then he would prefer sort 2 over sort 3 (the worst in the given example).

Each senator tries to make the choice meet his own preferences as much as possible provided that other senators follow their optimal strategy.

So, the preferences of this senator can be represented as sequence "1, 0, 2, 3" (here 0 means "giving up cabbage at all"). Your task is to find out the result of the vote for given preferences of senators.

Input

The first line of the input file contains two integers N, which stands for the number of senators, and K, the number of sorts of cabbage ($1 \le N, K \le 1000$). Then N lines follow that determine preferences for each senator. Each line contains a permutation of numbers $0 \dots K$, the sequence of each senator's preferences. Here, 0 means "give up eating cabbage at all".

Output

The output file must contain a single integer, the number of cabbage sort to win the vote, or "`Nothing`" if the vote fails.

Example

cabbage.in	cabbage.out
2 2 1 0 2 2 0 1	Nothing

Problem C. Senseless Cipher

Input file:	`cipher.in`
Output file:	`cipher.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

Once Prof. Unnamed was feeling bored; to amuse himself he invented a curious way of text ciphering. He used a number of definitions to describe it.

A *text* is a sequence of lower-case Latin letters, punctuation marks, spaces and line feeds. A *word* in a text is a nonempty sequence of lower-case Latin letters which is maximal in a sense that the symbols before and after the word, if present, are not lower-case Latin letters. A *punctuation mark* is one of the following: '`!`', '`?`', '`,`', '`.`', '`:`', '`;`', '`-`'. A *transformation rule* is a pair of words (w_1, w_2). The *application* of rule (w_1, w_2) towards word w_1 is the replacement of word w_1 with w_2. *Senseless text transformation* is a sequential application of one of the possible rules (possibly different) towards each word of a given text.

And, finally, senseless ciphering is a sequential application of exactly K senseless text transformations towards a given text, each transformation using the same set of rules.

Professor ciphered a text, and then wondered if any mistakes happened during ciphering, and if there was at least one possible source text.

You are facing a slightly more complex problem: to find out the number of possible variants of the source text. Since the number may be huge, the remainder after division by 1234567891 will suffice. If there is no source text that can be transformed into the given text using K transformations, assume 0 as the answer.

Input

The first line of the input file contains two integers: number of rules N ($0 \le N \le 10\,000$) and number of transformations K ($0 \le K \le 10^9$).

Then N lines follow, each containing a rule definition, i.e. a pair of words (w_1, w_2). Each word is a nonempty sequence of lower-case Latin letters. The length of each word here will not exceed 20 characters. The set of rules uses at most 100 different words.

After that, starting with a new line, the text is given expanding till the end of file. The text is no longer than 100 000 characters. Text can contain lower-case Latin letters, punctuation marks, spaces and line feeds.

Output

Output a single number, the number of possible variants for the source text modulo 1234567891.

Examples

cipher.in	cipher.out
3 1 b a c a a c a	2
5 6 a b b c c d d e e f f	0
6 1000 one moo moo moo yes moo word moo hello moo world moo moo, moo! moo moo moo, moo? moo, moo. moo moo moo. moo, moo moo moo! moo...	122276521

Problem D. MiniMaxFlow

Input file:	`flow.in`
Output file:	`flow.out`
Time limit:	10 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

Let $G = (V, E)$ be a directed network defined by a set V of n nodes (vertices) and a set E of m directed arcs (edges). We associate an integer number b_i with each node $i \in V$. You have to solve the following problem given in terms of optimization model:

Minimize $z(x) = \max\limits_{(i,j)\in E} x_{ij}$,

subject to

$$\sum_{\{j:(i,j)\in E\}} x_{ij} - \sum_{\{j:(j,i)\in E\}} x_{ji} = b_i \qquad \text{for all } i \in V,$$

$$x_{ij} \text{ is nonnegative integer for all } (i,j) \in E.$$

Network G doesn't contain loops (it means that $(i,i) \notin E$ for all $i \in V$) and between any two vertices $i \in V$ and $j \in V$ there can be at most one directed arc $(i,j) \in E$ and at most one directed arc $(j,i) \in E$.

Input

On the first line of the input file there are two integers n and m. Assume that $2 \le n \le 10\,000$ and $1 \le m \le 500\,000$.

The second line contains the sequence of integers b_i $(i = 1, 2, \ldots, n)$ separated by single spaces. $|b_i| \le 10^6$ for all $i = 1, 2, \ldots, n$.

The following m lines give information about graph structure: each line contains two numbers i and j separated by a single space. These numbers mean that the directed arc (i,j) is in E.

Output

Output the optimal value of $z(x)$. It is guaranteed that a solution exists.

Examples

flow.in	flow.out
4 5 10 0 0 -10 1 2 2 4 1 3 3 4 1 4	4
4 3 10 -1 -7 -2 1 2 1 3 3 4	9

Problem E. Grid

Input file:	`grid.in`
Output file:	`grid.out`
Time limit:	6 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

A *grid graph* (known under different names, such as *product lattice graph*, or $P_n \times P_m$ graph) is the graph whose vertices correspond to the points in the plane with integer coordinates, x-coordinates being in the range $1, \ldots n$, y-coordinates being in the range $1, \ldots, m$, and two vertices are connected by an edge whenever the corresponding points are at distance 1. For example, $P_1 \times P_1$ graph consists only of one vertex. $P_2 \times P_2$ graph consists of a single cycle with 4 vertices.

A *cycle* in a graph is a sequence of vertices such that from each of its vertices there is an edge to the next vertex in the sequence such that the first vertex and the last vertex are the same. Note that the choice of the first vertex in a cycle is arbitrary. A cycle with no repeated vertices aside from the first/last vertex is a *simple cycle*.

You are given $P_n \times P_m$ product graph. Find out how many simple cycles with 4 or more vertices it contains.

Input

The input contains two positive integers: n and m. Assume that $m \leq 8$ and $m \cdot n \leq 4000$.

Output

Output the required number of simple cycles on the first line.

Examples

grid.in	grid.out
1 1	0
1 2	0
3 1	0
2 2	1
2 3	3
2 4	6
3 3	13
8 8	603841648931

Problem F. Polymorph

Input file:	`polymorph.in`
Output file:	`polymorph.out`
Time limit:	3 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

A polymorphic virus is a type of computer viruses that are able to "mutate", that is, to alter their own code when infecting the target executable file. A virus alters its code to lower the risks of being detected by anti-virus software.

Consider the simplest option to implement a polymorphic virus. The major part of the program, namely, the target code and the body of the virus are encrypted using a random key. The entry point of the program is a special decryptor that has the decryption key built into its code. The decryptor itself is "littered" by the following algorithm: a random command is inserted several times at a random position in the decryptor code in such a way that this command does not alter the behavior of the original algorithm.

Let us be more formal. The input file contains the description of the decryptor written in pseudo-assembly language. The decryptor consists of three optional sections: *INIT*, *DECRYPT*, *DONE*. Each section comprises a number of operations executed sequentially, one by one. The *DECRYPT* section is enclosed in a program loop (using the `loop` command, see below) that will be executed at least two times. The `loop` may be absent — in this case the program consists of a single *INIT* section.

```
<INIT>
@:
<DECRYPT>
loop @, <mr>
<DONE>
```

In the imaginary hardware architecture under consideration, the commands may operate on memory, registers and numeric constants. All numeric constants in the program code are substituted with a '`?`' character. There are two types of memory addressing: absolute (`[?]`) and relative addressing (`[<some register>]`). In the latter case the address is the value that is stored in the given register. Each register is denoted with a lower-case Latin letter.

The table below lists all possible commands (`<mr>` stands for "memory or register").

Operation	*Meaning*
`mov <mr>, <mr>`	Move the value of the second argument to the first argument
`mov <mr>, ?`	Move the value of the numeric constant to the first argument
`swap <mr>, <mr>`	Swap the values of arguments
`op <mr>`	Perform a certain operation on the argument and store the result in the argument
`op <mr>, <mr>`	Perform a certain binary operation on the pair of arguments and store the result in the first argument
`op <mr>, ?`	Perform a certain binary operation on the first argument and the constant and store the result in the argument
`loop @, <mr>`	Perform an unary operation on `<mr>` and go to label `@` if the value of `<mr>` satisfies a certain condition; else go to the next command

Code "littering" is performed in the following way: a random command (with its arguments) is selected, and then this command is inserted at an arbitrary position in any section. The commands are inserted independently. Command being inserted will not alter the behavior of the algorithm under any conditions. In particular, "litter" commands may not work with memory.

Note also that:

- there is at least one "non-litter" command in the program;
- there is at most one loop in the program, and it may only surround the *DECRYPT* section; loop command cannot be "litter".

Your task is to remove the maximum number of "litter" commands (i.e. commands that do not alter the behavior of the decryption algorithm).

Input

The first line of the input file contains a single integer $N \leq 100\,000$, which stands for the number of lines in the decryptor code.

The following N lines contain the code of the decryptor. Each line is either a command or an "`@:`" label (the latter may appear at most once and it will precede the `loop` command).

Output

The first line of the input file should contain an integer M — the number of lines in the program after "litter" commands have been removed. Then the code of the "clean" program should follow, using the same notation as in the input file.

Examples

polymorph.in	polymorph.out

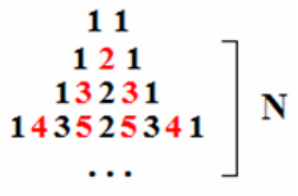
9
mov a, [?]
mov c, ?
mov d, ?
@:
op [d], a
mov c, a
swap b, c
loop @, d
mov d, ?
	5
	mov a, [?]
	mov d, ?
	@:
	op [d], a
	loop @, d

7
mov a, ?
mov c, ?
mov d, ?
op [d], a
mov c, a
swap b, c
mov d, ?
	3
	mov a, ?
	mov d, ?
	op [d], a

Problem G. Interesting Sequence

Input file:	`sequence.in`
Output file:	`sequence.out`
Time limit:	3 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

Let's build a sequence. Initially write two ones ("*1 1*"). Then write their sum ("*2*") between them. And so on, write sum of the each two neighboring numbers between them on each iteration. N iterations have been executed. Your task is to determine how many times the number N will be present in the resulting sequence.



Input

Input contains one integer K ($1 \le K \le 20$) on the first line — number of tests for this problem.

Each of the next K lines contains one integer N ($1 \le N \le 10^{14}$) — number of iterations in the corresponding process.

Output

Output should contain K numbers, one number per line — answers for each test.

Example

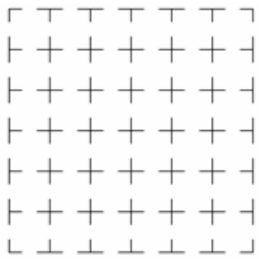
sequence.in	sequence.out
2 3 4	2 2

Problem H. 1000 Wives and One Sultan

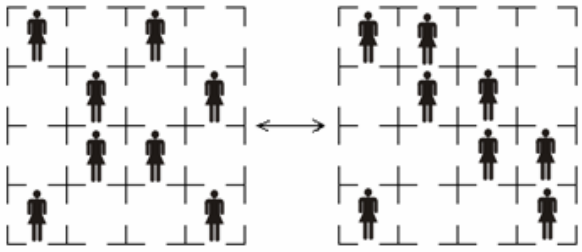
Input file:	`sultan.in`
Output file:	`sultan.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

Long ago in a beautiful country of Persia, a rich and wise sultan lived. As all respected sultans, he held a large harem. And according to an old Persian tradition, a garem should contain an even number of wives. According to it, the sultan had $2N$ wives. But the wives permanently argy-bargied with each other and therefore hindered sultan's lying and meditating of world's imperfection. Then a brilliant idea came to his head. He revised that an other palace contains a huge apartment of a square form, consisting of $N \times N$. Each room was connected with adjacent ones by aisles as is shown on the first picture.

So, from every room, it is possible to see all people in the same vertical and horizontal rows.



Picture 1



Picture 2

By means of hypnosis, sultan managed to suggest to his wives not to be crazy when they seeing not more than two others. And he arranged his wives in such a way that each vertical and horizontal row held no more than two wives. Peace and order came to the palace. But just few days after that, the sultan's peace was shattered. He couldn't succeed in calculating the number of non-equivalent ways in which all wives could be arranged. He treats two ways as equivalent if one can be transformed to another by means of some columns' and rows' transpositions as is shown on the second picture. The sultan considers wives to be equal.

Write a program that will help the sultan to solve the problem.

Note that number of nonequivalent locations may be too large, but sultan is not afraid of large numbers and takes them modulo 1024.

Input

Input file contains one integer N — the side of the apartment where $2 \le N \le 2000$.

Output

The output must contain one integer — the number of nonequivalent arrangements of wives modulo 1024.

Examples

sultan.in	sultan.out
2	1
11	14

Problem I. Yet Another Lucky Ticket

Input file:	`ticket.in`
Output file:	`ticket.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

The universe is facing yet another economic recession, and reactions to it differ a lot. For example, intergalactic passenger transport agency has decided to increase the fares for superlight shuttle travels. However, foreseeing the peoples indignation they moved forward a special offer. For each vehicle on route number N, a special class of tickets was determined, the so-called *N-lucky tickets*. Any passenger who happens to get an N-lucky ticket gets a free ride!

Ticket number $a_{m-1}a_{m-2}\ldots a_1a_0$ is N-lucky if all elements $s_i = \sum_{\substack{0\le j<m,\\ i\equiv j (mod N)}} a_j$ are mutually equal for $0 \le i < m$. Tickets are numbered sequentally using natural numbers and starting from 1.

Student Vassily was going home to Earth from the university on planet Chiron, Alpha Centauri. As usual, he took shuttle route number N and got a ticket (perhaps, for free). In order to kill some time, he decided to find out the number of the next N-lucky ticket. As you may have guessed your task is to help him solve this stiff problem.

Input

The first line of the input file contains the route number N ($1 \le N < 2^{31}$).

The number T of the ticket Vassily has got goes in the second line ($1 \le T \le 10^{100\,000}$). Number T is given without leading zeroes.

Output

Write the number of the closest N-lucky ticket following ticket number T to the output file omitting the leading zeroes.

Examples

ticket.in	ticket.out
3 123	222
2 101499	101530

Problem J. Wise Men

Input file:	`wisemen.in`
Output file:	`wisemen.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

Once N wise men gathered to find out who was the wisest. No matter how long they argued, they were unable to come up with a fair way to solve their problem. They finally summoned a cunning demon who offered the following.

At first the demon took B black and W white labels and showed these to the wise men. After that he gave each man a unique number and stuck two labels on everyone's forehead. Each man had not seen any of his labels and was unable to do so, while the rest could see them. The demon might have kept some labels (if twice the number of men is less than the total number of labels), but again, nobody had seen or could see this remainder.

Then the demon started querying the wise men in turn (the first one, the second, ..., the N-th, then the first again and so on). He asked the same question: "*Do you know the colors of labels on your forehead?*"

The answers appeared to be as follows.

Wise man #1, question #1: "*No, I don't know*".

Wise man #2, question #2: "*No, I don't know*".

...

Wise man $\#((Q-1) \bmod N + 1)$, question $\#Q$: "*Yes, I do know that!*"

The wise man who had number $(Q-1) \bmod N + 1$ was claimed to be the wisest of the meeting.

You will be given the values of N, B, W, Q (Q is the number of question for which a positive answer was given). Your task is to check if such a scenario is possible and, if so, output the label colors for each of the wise men. If several solutions exist, output the minimum result with respect to lexicographical order (a result is a correctly formatted output string, see below).

Note that each wise man is wise enough to determine color of labels on his forehead if it's possible.

Input

The first line of the input file contains an integer N ($1 \le N \le 9$). The second line contains two integers B and W ($0 \le B, W \le 100$, $B + W \ge 2N$). The integer Q is given on the third line ($1 \le Q \le 1000$).

Output

If the scenario described above is impossible for given input data, write "`Impossible`" (without quotation marks) to the output file. Otherwise, output a string consisting of N space-delimited character pairs denoting the label colors for each man: "`WW`", "`BB`", or "`BW`".

Examples

wisemen.in	wisemen.out
3 4 4 1	BB WW WW
1 1 1 2	Impossible

Problem K. Magic Power

Input file:	`wizard.in`
Output file:	`wizard.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Output limit:	32 megabytes

Consider a sequence of powers of 3: 3, 9, 27, 81, 243, 729, 2187, 6561, ... From these numbers, take the subsequence of numbers for which their decimal presentation starts with digit '9' and define them as $3^{f(1)}$, $3^{f(2)}$, $3^{f(3)}$, ...

Your task is to find $f(k)$ for a given k.

Input

The first line contains integer N — the number of testcases ($N \le 100$).

Each of the following N lines contains k_i ($1 \le k_i \le 10^7$).

Output

Output file should contain N lines, each of them containing $f(k_i)$.

Example

wizard.in	wizard.out
2 1 2	2 23