Problem A. Blocking Buffer

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 megabytes

New microarchitecture for parallel programming has a pipe buffer to organize the data exchange between two execution threads. The buffer is circular and can contain no more than l bytes. At the beginning of the program execution the buffer is empty. The first thread sometimes tries to write to the buffer, and it always uses blocks of size w bytes. The second thread sometimes reads from the buffer in blocks of size r bytes each. All write operations put the new data at the end of the pipe buffer and all read operations take the data from the beginning of the pipe buffer, so empty positions always form one continuous segment.

Both write and read operations are blocking. It means that, if the first thread tries to write the block of data to the buffer but there is not enough space there, the thread stops and waits until there will be at least w empty bytes in the buffer. Similarly, if the second thread tries to read the block of data from the buffer but there are less than r bytes of data left, the thread stops and waits until there will be enough new data in the buffer.

In this particular problem, a *deadlock* is a situation when the first thread is blocked because there is not enough space in the buffer to fit a new block of size w, and the second thread is blocked because there is not enough data in the buffer to read a new block of size r.

Your goal is to determine whether there exists such a sequence of reads and writes from two threads that will result in a deadlock. Note that write operations are only performed by the first thread and read operations are only performed by the second thread.

Input

The only line of input contains three integers l, r and w ($0 < l, r, w \leq 10^{18}$, $r \leq l$, $w \leq l$).

Output

If there exists a sequence of reads and writes that result in a deadlock, print "`DEADLOCK`" (without quotes) in the only line of the output. Otherwise, print "`OK`" (without quotes).

Examples

standard input	standard output
5 3 4	DEADLOCK
5 2 3	OK

Note

One of the ways to obtain the deadlock in the first sample is:

1. The first thread writes a block of size 4 to the buffer.
2. The second thread reads a block of size 3. There is 1 byte of data left in the buffer.
3. The first thread writes a block of size 4. The buffer is full now.
4. The second thread reads a block of size 3. There are 2 bytes of data left in the buffer.
5. The second thread tries to read a block of size 3, but there is not enough data in the buffer, so the thread is blocked.
6. The first thread tries to write a block of size 4, but there it not enough free space in the buffer, so the thread is also blocked. The deadlock just happened.

Problem B. Benches

Input file:	*standard input*
Output file:	*standard output*
Time limit:	0.5 seconds
Memory limit:	64 mebibytes

The city park of IT City contains n east to west paths and n north to south paths. Each east to west path crosses each north to south path, so there are n^2 intersections.

The city funded purchase of five benches. To make it seems that there are many benches it was decided to place them on as many paths as possible. Obviously this requirement is satisfied by the following scheme: each bench is placed on a cross of paths and each path contains not more than one bench.

Help the park administration count the number of ways to place the benches.

Input

The only line of the input contains one integer n ($5 \le n \le 100$) — the number of east to west paths and north to south paths.

Output

Output one integer — the number of ways to place the benches.

Examples

standard input	standard output
5	120

Problem C. Index of Permutation

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

You are given a permutation of all integers from 1 to n inclusive. Find its position in the array of sorted (lexicographically) permutations of length n (its index).

Input

The first line of the input contains a single integer n ($1 \le n \le 12$) — the length of the permutation. The second line contains n distinct integers from 1 to n that define the permutation itself.

Output

Print one integer — the index of the given permutation.

Examples

standard input	standard output
3 2 1 3	3

Problem D. Chinese Theorem

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

Find integer solution for the following system of equations.

$$\begin{cases} x \equiv a \pmod{n} \\ x \equiv b \pmod{m}, \end{cases}$$

where n and m are guaranteed to be co-prime. If there are several valid x you should print minimal one.

Input

The input contains four integers a, b, n and m ($1 \le n, m \le 10^6$, $0 \le a < n$, $0 \le b < m$).

Output

Print minimal possible valid x. It is guaranteed that at least one solution exists.

Examples

standard input	standard output
1 0 2 3	3
3 2 5 9	38

Problem E. Regular Bracket Sequences

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	64 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 $s = t_1t_2$.

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 F. Sieve It!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

For a positive integer n define following functions:

- $d(n)$ — minimal divisor of n greater than 1, put $d(1) = 0$ by definition.
- $s_0(n)$ — number of different divisors of n.
- $s_1(n)$ — sum of all divisors of n.
- $\varphi(n)$ — totient Euler function, the number of integers k such that $1 \le k \le n$ and $GCD(n, k) = 1$.

Given integer n, calculate $\sum_{k=1}^{n} d(k)$, $\sum_{k=1}^{n} s_0(k)$, $\sum_{k=1}^{n} s_1(k)$ and $\sum_{k=1}^{n} \varphi(k)$.

Input

The only line of input contains integer n ($1 \le n \le 10^7$).

Output

Print four space-separated numbers: $\sum_{k=1}^{n} d(k)$, $\sum_{k=1}^{n} s_0(k)$, $\sum_{k=1}^{n} s_1(k)$ and $\sum_{k=1}^{n} \varphi(k)$.

Examples

standard input	standard output
10	28 27 87 32

Problem G. Cabbages Under Hyperbola

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

Farmer John has bought a patch of field. The patch consists of several cells on the rectangular grid. Farmer John introduced coordinate system with axes aligned to grid lines. For any integers x and y, a cell (x, y) belongs to Farmer John's field if $x > 0$, $y > 0$ and $xy \leq n$.

Farmer John wants to choose a rectangular piece of his field and plant it with cabbages. Borders of the piece must lie on the grid lines. Of course, all cells inside the chosen piece must belong to Farmer John's field, and the piece must have positive area.

Help Farmer John count the number of different pieces he can plant with cabbages.

Input

The only line of input contains one integer n ($1 \leq n \leq 10^{15}$).

Output

Print one integer — the number of ways to choose a rectangular piece of field, modulo $10^9 + 7$.

Examples

standard input	standard output
2	5
4	23

Problem H. Win or Freeze

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

This problem is borrowed from Codeforces.

You can't possibly imagine how cold our friends are this winter in Nvodsk! Two of them play the following game to warm up: initially a piece of paper has an integer q. During a move a player should write any integer number that is a *non-trivial* divisor of the last written number. Then he should run this number of circles around the hotel. Let us remind you that a number's divisor is called *non-trivial* if it is different from one and from the divided number itself.

The first person who **can't make a move wins** as he continues to lie in his warm bed under three blankets while the other one keeps running. Determine which player wins considering that both players play optimally. If the first player wins, print any winning first move.

Input

The first line contains the only integer q ($1 \le q \le 10^{13}$).

Please use the `%lld` specificator to read or write 64-bit integers in C++.

Output

In the first line print the number of the winning player (1 or 2). If the first player wins then the second line should contain another integer — his first move (if the first player can't even make the first move, print 0). If there are multiple solutions, print any of them.

Examples

standard input	standard output
6	2
30	1 6
1	1 0

Note

Number 6 has only two non-trivial divisors: 2 and 3. It is impossible to make a move after the numbers 2 and 3 are written, so both of them are winning, thus, number 6 is the losing number. A player can make a move and write number 6 after number 30; 6, as we know, is a losing number. Thus, this move will bring us the victory.

Problem I. Shall We Play a Game?

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

This is an **interactive** problem.

1804. Vice-president of the United States of America Aaron Burr challenged gubernatorial race candidate Alexander Hamilton for writing several offending pamphlets.

But Burr is sane enough to understand that, even if he kills Hamilton, he will lose his reputation and ruin his career. So, the enemies decided to simply play a game. To make everything more fair they decided to play it g times.

Each game starts with Hamilton thinking up a positive integer n, and after that Burr tries to guess it. For any positive integer x Burr may ask Hamilton about fraction of numbers between 1 and n inclusive that are divisible by x. In the other words, when asks a question about x, he receives the value of

$$\frac{\left\lfloor \frac{n}{x} \right\rfloor}{n}.$$

Important detail is that Hamilton reports the answer to Burr as an **irreducible** fraction (here $\lfloor r \rfloor$ denotes the integral part of r).

Help Burr find the answer using some certain number of queries.

Interaction format

At the beginning your solution receives a single integer g ($1 \le g \le 1000$), the number of games Hamilton and Burr are going to play.

For each test the number q ($6 \le q \le 60$) is fixed, the maximum number of queries in a single game. It is guaranteed that q queries are enough to solve the problem. This number is not available for your program, but it is mentioned in a scoring table below. If your program performs more than q queries, it receives the "Wrong answer" verdict.

In order to perform a query you should output a line "`X` t" ($1 \le t \le 10^{18}$) where t is a positive integer for which you want to know the value of

$$\frac{\left\lfloor \frac{n}{t} \right\rfloor}{n}$$

As a result for a query your program receives two integers a and b, the numerator and denominator of an irreducible fraction or number -1 in case you exceed the query limit.

If your program determines the number n, it should output a line "`N` t" where t is the supposed answer ($1 \le t \le 10^{18}$). If the answer is correct, solution receives a line "`Correct`", otherwise it receives a line "`Wrong`".

After that the next game starts (if any of them left), otherwise solution should exit.

Note that in case you read -1 or line "`Wrong`" you **must** exit your programm immediately. Otherwise the verdict for your solution may be wrong, in particular you may receive Run-time error or Time limit exceeded!

It is guaranteed that in each case the n numbers are fixed at the beginning of the game and they are not changed depending on your queries.

Examples

standard input	standard output
2 1 2 3 10 1 5 1 5 1 10 1 10 0 1 Correct 1 1 0 1 Correct	 X 2 X 3 X 5 X 4 X 6 X 10 X 11 N 10 X 1 X 2 N 1

Notes

In the first sample case $g = 2$. The example queries are listed above, using results of these queries player may understand that in the first game the answer is 10, and in the second game the answer is 1.

Strictly follow the output format. After printing anything make sure to print a new line and to flush the output buffer, in order to do that you may use `flush(output)` for Pascal/Delphi, `fflush(stdout)` or `cout.flush()` for C/C++, `sys.stdout.flush()` for Python, `System.out.flush()` for Java.

Problem J. Famil Door and Brackets

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

As Famil Door's birthday is coming, some of his friends (like Gabi) decided to buy a present for him. His friends are going to buy a string consisted of round brackets since Famil Door loves string of brackets of length n more than any other strings!

The sequence of round brackets is called *valid* if and only if:

1. the total number of opening brackets is equal to the total number of closing brackets;
2. for any prefix of the sequence, the number of opening brackets is greater or equal than the number of closing brackets.

Gabi bought a string s of length m ($m \leq n$) and want to complete it to obtain a valid sequence of brackets of length n. He is going to pick some strings p and q consisting of round brackets and merge them in a string $p + s + q$, that is add the string p at the beginning of the string s and string q at the end of the string s.

Now he wonders, how many **pairs** of strings p and q exists, such that the string $p + s + q$ is a valid sequence of round brackets. As this number may be pretty large, he wants to calculate it modulo $10^9 + 7$.

Input

First line contains n and m ($1 \leq m \leq n \leq 100\,000, n - m \leq 2000$) — the desired length of the string and the length of the string bought by Gabi, respectively.

The second line contains string s of length m consisting of characters '`(`' and '`)`' only.

Output

Print the number of pairs of string p and q such that $p + s + q$ is a valid sequence of round brackets modulo $10^9 + 7$.

Examples

standard input	standard output
4 1 (	4
4 4 (())	1
4 3 (((	0

Note

In the first sample there are four different valid pairs:

1. $p =$"(", $q =$"))"
2. $p =$"()", $q =$")"
3. $p =$"", $q =$"())"
4. $p =$"", $q =$")()"

In the second sample the only way to obtain a desired string is choose empty p and q.

In the third sample there is no way to get a valid sequence of brackets.

Problem K. Matchings

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

Let us consider a bipartite graph with $n + m$ vertexes. The left set contains n vertexes. Every vertex has a integer written in it $a_1, a_2, \ldots, a_n$. The right set contains m vertexes. Every vertex has a integer written in it $b_1, b_2, \ldots, b_m$. i-th vertex of left set and j-th vertex of right set are connected by edge only if $a_i \geq b_j$.

Count the number of different matchings containing m edges in this graph.

Input

First line contains two integers n and m, $1 \leq n, m \leq 10^5$. Second line contains integers $a_1, a_2, \ldots, a_n$, $1 \leq a_i \leq 10^6$ separated by a space. Third line contains integers $b_1, b_2, \ldots, b_m$, $1 \leq b_i \leq 10^6$ separated by a space.

Output

Output one integer - answer to the problem modulo $10^9 + 7$.

Example

standard input	standard output
3 2 7 3 4 1 5	2

Problem L. Beautiful board

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

You have a $n \times n$ board and several checkers of different colors. You want to place all of the checkers on the board in such a way that no cell contains more than one checker.

Output the number of different possible placements modulo 1234567891. Two placements are equal if you can get one from the other by rotating the board. Note that checkers of the same color are indistinguishable. If you have more checkers than the number of cells on the board, there are no possible placements, so you should output 0.

Input

First line of input contains two integers n and m ($1 \le n \le 10^5$, $1 \le m \le 50$).

Second line contains m numbers between 1 and 10^5; i-th number is number of checkers of color i.

Output

Print the number of possible placements modulo 1234567891.

Examples

standard input	standard output
1 1 1	1
2 2 1 2	3
3 4 4 2 1 3	0

Note

You are *not allowed* to flip the board.

Problem M. Next Combination

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

Consider a set of all integers from 1 to n inclusive. Consider all its subsets of size k as increasing sequences of integers and order them lexicographically. You are given one subset of size k, print a subset that goes after the given one in the arrangement.

Input

The first line of the input contains two integers n and k ($1 \le k \le n \le 50$). The second line contains an increasing sequence of k integers that define the subset you need to process.

Output

Print the subset of size k that goes after the given one in lexicographic order. If there is no such subset print 0.

Examples

standard input	standard output
6 4 1 4 5 6	2 3 4 5
6 2 5 6	0