Problem A. DIY Radar

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

This is an interactive problem.

Pin is an avid inventor who creates awesome mechanisms and robots from junk. While constructing his new junkyard worker robot, Pin made a mistake of activating the robot's AI earlier than usual, which made the robot unpredictably run away into the junkyard.

Pin wants to go into the junkyard to pick the robot up, but the area is just too big. The robot ran away before having a GPS tracker installed, so Pin has no information about the robot's exact coordinates. If only there was a way to get a rough estimate of where the robot could be...

Pin remembered the robot had an autonomous radio repeater in one of its parts. If Pin would send a specific radio signal, the reply would give him at least some important information: the distance between him and the robot, which can be used for triangulation.

Pin's junkyard is a huge square with its opposite corners lying at points $(0, 0)$ and $(100, 100)$, and small paths run parallel to square sides between junk piles, intersecting at integer points. Robot's AI is programmed to situate himself in a random path intersection at the junkyard (including the junkyard's border) and work around there. Unfortunately for Pin, a security algorithm in the rogue robot complicates the task. Every time the robot detects a searching radio signal, it would move to an adjacent intersection. Pin decided that it would be easier and more sensible to find the intersection where the robot initially was, and then search for it through a number of surrounding intersections.

Pin does not want to step into the junkyard before knowing where the robot is, so he will use his radio from the area around it. But he will not go beyond his privately owned area, which contains every point that is no further than 1000 vertically or horizontally from the junkyard's border.

Interaction Protocol

Your solution can print lines in the following formats:

1. "`Q` X Y": ask what is the distance between the point (X, Y), which lies outside the junkyard but within the privately owned area, and the robot. The system prints a single line with an integer R^2: the requested distance, squared. After that, the robot immediately moves to an adjacent intersection.
2. "`A` X Y": claim that the robot was at the point (X, Y) initially, before the measuring started. After printing this line, the solution must immediately terminate gracefully.

The total number of printed lines of both types must not exceed 5.

Example

standard input	standard output
25 4	Q 101 101 Q 97 101 A 97 98

Note

The example shows a possible interaction between a solution and the testing system. Blank lines in the example are added there only to show the order of events.

Do not forget to flush the buffer after each line you output! Here are some functions which do it in different languages:

- **C++:** `fflush(stdout)` or `cout.flush()`

- **Python:** `sys.stdout.flush()`
- **Java:** `System.out.flush()`

Problem B. Word Squared

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

Given a permutation of numbers from 1 to n, find a square matrix conforming to the following rules:

1. The matrix should include only numbers from the permutation;
2. The given permutation should occur in every row of the matrix as a contiguous subsequence, read from left to right;
3. The given permutation should occur in every column of the matrix as a contiguous subsequence, read from top to bottom;
4. The matrix size is the smallest possible.

Input

The first line of input is a positive integer $n \le 500$.

The second line of input consists of n space-separated integers: the permutation itself.

Output

The first line of output should be an integer m: the size of the matrix. The next m lines should list m consecutive rows of the matrix. Each of these lines should contain m integers separated by spaces: the values in the corresponding row.

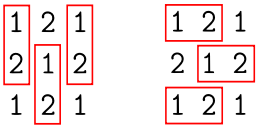
The size m should be minimum possible. If there are several possible answers, print any one of them.

Example

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

Note

Here is where the permutation occurs in the matrix from the example:

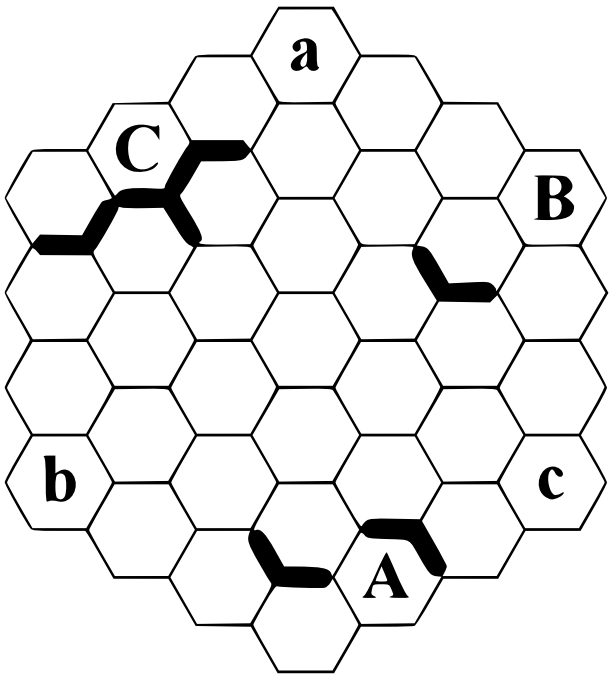


Problem C. Quoridor

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

"Quoridor" is a strategy board game for two or four players. It is played on a square game board consisting of square cells. Each player has a pawn and a set of walls, each wall two cells long. Pawns can stand inside the square cells, and walls can be put between cells. The goal of the game is to move your pawn across the board before all the other players do. Pawns move only to adjacent cells and can not move through walls. In case a pawn is moving to a cell already occupied by another player's pawn, the cell can be "skipped over" by moving further to an adjacent cell. Walls can not overlap or cross, but they can touch. Walls should be placed in such a way that all the players always have an option to cross the board: you can not block a player from finishing the game completely.

Three players play a new version of the game for three players. The board in the new game is hexagonal, with hexagonal cells. Player pawns start from three board corners such that no two corners are neighbors. The objective of each player is to cross over to the opposite corner. Walls here also have the length of two cell sides, but now they are bent because of the new shapes of the cells. No portion of a wall can run along the edge of the board. Here is an example of the game being played:



Uppercase characters are players' pawns,
lowercase characters are their respective goals

Player `A` has to move now. They don't want to move their pawn yet, so they consider all the possible ways they can place a wall on the board. How many such placements are there?

Input

The first line of input contains an integer n: the number of cells along one side of the hexagonal board, $2 \le n \le 250$.

The rest of the input lines are depiction of the current game situation in a form of ASCII art. The number of lines in the board description is equal to the number of characters in each line, and is $4n - 1$. Cell borders are drawn with characters "\", "/", and "_". Walls are depicted by characters "`@`" in place of cell borders. Uppercase letters "`A`", "`B`", and "`C`" depict player pawns. Lowercase letters "`a`", "`b`", and "`c`" are those players' respective goals. Each of the six letters appears on the board exactly once. The lowercase letters are placed according to the rules of the game.

The remaining characters in the board description are all space characters. Note that most lines, but not all of them, start and end with a space character, and each line contains at least one non-space character.

Output

Print a single line containing a single integer: the number of possible ways for player `A` to place a wall on the board.

Examples

standard input	standard output

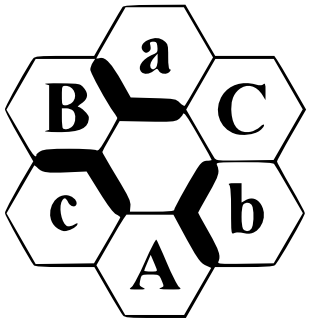
2
  _
 _/a_
/B_/C\
_/ _/
/c_/b\
_/A_/
  _/
	18

2
  _
 _/a_
/B@@/C\
\@/ _/
/c@_@b\
_/A@_/
  _/
	0

Note

In the first example, wall can be put to any of the two available positions adjacent to any of the six border cells, and at the six available positions adjacent to the center cell, for a total of $2 \cdot 6 + 6 \cdot 1 = 18$ placements.

The second example presents a situation in which walls can not be put anywhere without blocking some of the players from winning.



Problem D. Game X

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Sometimes one loves board games so much they start inventing their own games. Bob came up with a game with such overcomplicated rules that we will provide only a small portion of them.

Each of the $n+1$ players has a *power change* card. Card values are positive and negative integers which are unique by absolute values (for example, if there is a card with value -4, there cannot be a card with value 4). There is no card with a value of zero.

One of the players gets a *target* token, which means that the other n players have to *attack* them.

Attacking in this game goes like this: two random players simultaneously reveal their power change cards, and the product of their values is added to the target's power value (oh, we forgot to mention that each player has a power value). Since the cards can have both positive and negative values, the product can turn out positive or negative as well, thus the power value could increase or decrease. Of course, the target player wants to increase their power value.

On the new game's test run, Bob got the target token. Given that he invented this game, Bob vaguely remembers there should be exactly k possible unordered pairs of attacking players such that their power change card values had positive **sums**, which he learned before he switched the rules from addition to multiplication a long time ago. Could that help with finding the largest possible number of unordered pairs of players that can increase Bob's power value now?

Input

The only line of input consists of two space-separated integers n and k ($2 \le n \le 10^9$, $0 \le k \le 10^{18}$).

Output

If the input data is contradictory, which can only mean that Bob misremembered the number and there cannot possibly be exactly k unordered pairs of players with positive card value sums, output -1.

Otherwise, the only line of output should contain an integer which is the largest number of unordered pairs of players that can increase Bob's power value.

Examples

standard input	standard output
2 1	1
30 374	354

Problem E. 5-Path

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

You are given a list of edges of an undirected graph. There are two special nodes in the graph: a and b. Find the minimum size of a prefix of this list such that a graph represented by this prefix includes a simple path of 5 edges between nodes a and b.

Input

The first line of input contains two integers n and m: the number of nodes and the number of edges in the graph, respectively ($1 \le n \le 10^5$, $1 \le m \le 2 \cdot 10^5$).

Each of the following m lines contains two integers v_i and u_i which describe two endpoints of an edge ($1 \le v_i, u_i \le n$).

The last line contains two integers a and b: the numbers of special nodes ($a \ne b$, $1 \le a, b \le n$).

The graph has no multiple edges and no self-loops.

Output

If there is a simple path of 5 edges in the graph represented by the given edge list, output the answer to the problem. Otherwise, output -1.

Examples

standard input	standard output
10 9 1 6 6 9 9 10 10 3 3 2 2 7 7 5 5 8 8 4 6 7	6
10 10 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9 10 2 10 3 1 10	-1

Problem F. Nightmare

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

A car is traveling along the famously dilapidated roads of country N to get from point A to point B. It is known that the only road connecting these places has exactly n potholes, and the car can only withstand k potholes. In case there is just one more of them, the car stops dead and falls apart.

A pothole can be represented by a three-dimensional polyhedron with one of its faces lying on the horizontal Oxy plane, and the rest of the polyhedron being below the plane. The top face is the "hole" in the pothole. If a vertical plane is drawn in such a way that any of the top face's edges lies in this vertical plane, the whole polyhedron will lie in a single half-space. If viewed from above, with the top face removed, this polyhedron will have every point of its surface visible.

The car in question can be considered a rectangle on the Oxy plane, with its corners being its wheels. The car starts riding on a flat surface, without any potholes, and is always moving in a single predefined direction. The car hits a pothole if it gets any non-zero number of its wheels into the pothole. For each pothole, only the earliest hit is taken into account. Hitting only the edge of a pothole does not count as hitting it. For simplicity, the car is considered to always move in the Oxy plane, even when it hits a pothole.

How far could the car travel before it breaks down?

Input

The first line of input consists of a single integer k: the number of potholes the car can tolerate ($1 \le k \le 50$).

The next four lines describe the coordinates of the car's wheels on the Oxy plane. Each description has the form of "x y", where x and y are integers and do not exceed 10 000 by absolute value. Descriptions go in the following order:

1. Front right;
2. Front left;
3. Back left;
4. Back right.

The car is moving along the normal to a line passing through the two front wheels (like a regular car).

The next line of input contains an integer n: the number of potholes ($1 \le n \le 50$). The descriptions of potholes follow.

Each pothole is described in the following way. First goes a line with a single integer m_i: the number of vertices in the polyhedron which is a model for the i-th pothole ($4 \le m_i \le 300$). Each of the next m_i lines describes a single vertex. The description consists of three integers, x, y, and z: the coordinates of the vertex ($-10^4 \le x, y \le 10^4$, $-10^4 \le z \le 0$).

It is guaranteed that potholes do not intersect and do not touch.

Output

Output the distance the car will travel before breaking down. The answer will be accepted if the absolute error is at most 10^{-4}. In case the car never breaks down, output "oo" (two lowercase letters "o").

Example

standard input	standard output
2 0 0 -2 2 -4 0 -2 -2 3 5 1 1 0 3 3 0 2 2 -1 1 3 0 3 1 0 5 0 4 0 1 5 -1 0 6 0 2 6 0 2 4 0 5 6 6 0 6 4 0 4 4 0 5 5 -1 4 6 0	5.65685425

Problem G. String Transformation

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

Cody wrote down a string of English letters and an integer k. Now he wants to transform this string so that it would have exactly k "holes". A hole is a closed loop in a letter. For example, the letter "B" has two holes, and the letter "a" has one hole.

Cody can transform the string by choosing a letter and "incrementing" it, changing it into the next letter of the alphabet (for the last letter of the alphabet, the next one is the first letter of the alphabet). Cody can repeat this action as much as he wants with any letter in the string.

In Cody's understanding, the alphabet is this sequence of English letters:

AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz

All the letters Cody writes look exactly like the letters in the alphabet depicted above.

Naturally, Cody wants to perform as little increments as possible to achieve his goal. If the string cannot be transformed to contain the needed amount of "holes", Cody sees it and just doesn't bother to perform any transformations. Can you be as smart as Cody?

Input

The first line of input consists of two space-separated integers n and k: the length of the string and the required number of "holes" ($1 \le n \le 10^5$, $-10^{18} \le k \le 10^{18}$).

The second line is the string s Cody wrote down. The string s includes only uppercase and lowercase English letters.

Output

Output the number of increments needed to achieve the goal, or -1 in case it is impossible.

If the string can be transformed, the second line of output should be the resulting string. If there are several possible answers, print any one of them.

Example

standard input	standard output
5 3 zlEna	2 zlenB

Problem H. Employees

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

There are n employees in the company Yooglex. The i-th employee has to work t_i time units per day. These values are different for all employees. Yooglex work process is weird in such a way that at every moment in time only one employee can work, and the rest of them have to wait until he's done.

Yooglex office consists of a hall and a room. No more than k employees can be in the hall at the same time. Employees form a queue outside the office to enter the hall. Then the next algorithm applies:

1. While there are less than k employees in the hall and at least one employee in the queue, the front employee from the queue enters the hall.
2. If the hall is empty, the day is considered over and the room is locked.
3. Employees in the hall choose the one among themselves whose workload for the day is minimal.
4. The chosen employee from the hall enters the room, works his time and leaves the office for the day.
5. The process returns to step 1.

The board of Yooglex finally tasked themselves with evaluating their employees to assign sensible salaries. Two methods of evaluation have been coined.

The first one is quite simple. For each of $n!$ possible permutations of employee queues and each employee i, the time between the first employee in the queue entering the room and employee i exiting the room is calculated. For each employee, these times are summed over all the $n!$ queue permutations. The resulting sums are the final evaluation scores.

The second method is even simpler. For i-th employee and each queue permutation, consider all such employees j that $t_j > t_i$, but employee j is going to finish his job before employee i. The number of such employees j will be the score for employee i.

It has been decided that attempting to choose one of two methods is a fruitless task, and instead the third method has been suggested, which is simply a product of respective scores from the first two methods.

Can you calculate the third evaluation method scores quick enough?

Input

The first line of input consists of two integers n and k: the number of employees and the hall capacity, respectively ($1 \le n \le 300$, $1 \le k \le n$).

The second line consists of n integers $t_1, \ldots, t_n$: the workloads of each employee ($1 \le t_i \le 10^9$). All t_i are different.

Output

Output n integers: the evaluation scores for employees, listed in the same order as their workloads in the input. Since the scores can be very large, output them modulo $10^9 + 7$.

Examples

standard input	standard output
3 1 2 1 3	72 126 0
4 2 10 30 2 7	0 0 3564 2208

Problem I. Modulo-magic squares

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

A matrix of size $n \times n$ filled with integers from 0 to $m-1$ is called a *modulo-magic square modulo* m if there exists a constant C such that the following congruence relations hold:

$$\sum_{i=1}^{n} a_{i,j} \equiv C \pmod{m} \quad \text{for } j = 1, \ldots, n$$

$$\sum_{j=1}^{n} a_{i,j} \equiv C \pmod{m} \quad \text{for } i = 1, \ldots, n$$

$$\sum_{i=1}^{n} a_{i,i} \equiv C \pmod{m}$$

$$\sum_{i=1}^{n} a_{i,n-i+1} \equiv C \pmod{m}$$

In words, the sums of numbers in each row, column and both diagonals modulo m are equal.

Given n and m, find how many different modulo-magic squares modulo m of size n exist.

Input

The first line of input contains an integer T: the number of test cases ($1 \le T \le 100$). After that, T lines follow. Each of these lines contains two integers n and m: the size of the matrix and the modulo ($3 \le n \le 10^9$, $2 \le m \le 10^9$).

Output

For each test case, output the number of modulo-magic squares on a separate line. Since the answers can be very large, output them modulo $10^9 + 7$.

Example

standard input	standard output
1 3 2	8

Problem J. Count the Sequences

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

Given four integers, m, b, c, and n, calculate the number of integer sequences $x_1, \ldots, x_m$ such that:

- $0 \le x_i \le b^i - c$;
- $\sum_{i=1}^{m} x_i < n$.

Print the answer modulo 998 244 353.

Input

The first line of the input contains three integers, m, b, and c ($1 \le m \le 50$, $2 \le b \le 10^9$, $-b + 2 \le c \le b - 1$). The second line contains a large integer n ($1 \le n \le b^{m+1}$).

Output

Print one integer: the number of sequences modulo 998 244 353.

Example

standard input	standard output
2 3 1 5	12