Problem A. Airport Check-in

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

Pusheen decides to finally come to Japan and meet her friends. Pusheen arrives at the airport and she needs to check in for the flight. At the moment, n check-in counters are working, and there is a huge queue of passengers to each of them. Every counter has already been working for some time, and it is not clear when it started the process with the current passenger. It is known that each counter spends some fixed time to check in a passenger, but different counters may spend different amounts of time. It is known that this fixed time for each counter is real and comes from the uniform distribution from l to r.

Let's add a bit of formalism. For each counter, firstly, we choose a value t: the time that it spends for each passenger, a random real number uniformly distributed from l to r. After that, we choose the time that the counter needs to finish the process with the first passenger: the one that began the check-in process before Pusheen arrived. It is a random real number uniformly distributed from 0 to t. The choices for distinct counters are independent of each other.

Pusheen wants to check in for the flight as soon as possible, so she wants to choose the counter (and its queue) that works faster than the others. In hopes of achieving that, Pusheen chooses the counter that is the first to finish the check-in process with the current passenger.

What is the probability that Pusheen will choose the fastest counter this way?

Input

The first line of input contains one integer t, the number of test cases ($1 \le t \le 100$).

Each of the next t lines describes a single test case and contains three integers n, l, and r: the number of counters and the bounds on working time ($2 \le n \le 4$; $1 \le l < r \le 50$).

Output

For each test case, output the required probability on a separate line.

Your answer must be correct to within an absolute or relative error of 10^{-7}. Formally, let your answer be a, and the jury's answer be b. Your answer will be considered correct if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-7}$.

Example

standard input	standard output
2 3 4 5 2 1 10	0.362041935348 0.708657932673

Problem B. Beyond the Rescue

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

Riatla is a little-known member of the Assassins Order (well-known assassins, as you can guess, don't live long). After performing one mission, Riatla faces a problem: the streets of the city are flooded by guards, and he needs a new escape plan from the crime scene.

The city where Riatla is at the moment has a tree-like structure. There are n squares, connected with $n - 1$ two-way streets, and there is only one path between every two squares.

Riatla's associates find out that the guards patrol the city in a cyclic route: they choose a sequence of squares $v_1, v_2, \ldots, v_k$, and, after visiting square v_i, the guards head to square v_{i+1} (they may be not connected by a direct street, and then the guards go along the shortest path between them). After visiting square v_k, they go to square v_1, and start their route from the beginning.

Riatla is good at disguising. All the squares in the city are crowded, so the assassin can stay in a square without fear of being noticed by the guards even if the guards are on the same square. On the other hand, there are not many people on the streets, so if the assassin and the guards happen to be on the same street at any moment of time, the guards will shoot the assassin with bows regardless of the distance between them.

You know the plan of the city and how long it takes both the guards and the assassin to pass through every street. You should compute the minimal time that Riatla needs to spend to get from square s, where he is now, to square t, where his friends are waiting for him. As this time can be rather long, you only need to compute its remainder modulo 998 244 353.

Input

On the first line of input, you are given one integer t: the number of test cases. After that, t test cases are given.

The first line of a test case description contains two integers n and k: the number of squares in the city and the number of important squares on the guards' path ($2 \le n, k \le 200\,000$).

Each of the next $n-1$ lines contains four integers u_i, v_i, c_i, and d_i: the squares connected by streets and the times necessary for the guards and the assassin to walk through each street, respectively ($1 \le u_i, v_i \le n$; $1 \le c_i, d_i \le 10^8$). It is guaranteed that the given graph is a tree.

On the next line, you are given k integers a_i: the indices of important squares in the guards' path in the order in which the guards visit them ($1 \le a_i \le n$; $a_i \ne a_{i+1}$; $a_1 \ne a_k$).

On the next line, you are given two integers s and t: the starting square of the assassin and the last square of his path ($1 \le s, t \le n$; $s \ne t$).

It is guaranteed that both the sum of n and the sum of k in all the test cases don't exceed 200 000.

Output

For each test case, print one number on a separate line: the minimal time the assassin has to spend, modulo 998 244 353.

If the assassin can't get to his target without being spotted, print -1.

Example

standard input	standard output
2 5 4 1 2 1 4 2 3 1 4 3 4 1 4 4 5 10 10 1 4 1 5 1 4 2 2 1 2 2 1 1 2 1 2	16 -1

Problem C. Casino Cheating

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

This is an interactive problem.

Casino "Three Devices for Lumbering" has come up with a new gambling game which has immediately received much attention.

The rules of this game are very easy. Two players alternate their turns in the game: a visitor and a croupier. At the start of the round, the croupier has a whole chocolate bar, and the visitor goes first.

In his turn, each player takes any of the pieces of chocolate which belong to the opponent, cuts it into two pieces so that the bigger piece is at most twice as large as the smaller. After that, he takes one of these two pieces for himself and leaves the other one.

The casino decides not to investigate any optimal strategies in this game. Instead, the croupier in his turn chooses one of the visitor's pieces with equal probability, and takes as much as possible: $\frac{2}{3}$ of this piece. The randomness of the croupier's turns should convince the players that the croupier will make random and wrong turns.

The game lasts for a fixed odd number of turns: as the visitor starts without any chocolate at all, and croupier has a whole chocolate bar, it is guaranteed by the rules that the visitor makes the first and the last turns.

The cost of participation in one round is rather high: the player must pay 0.55 chocolate units, or c.u., which will not be refunded after the game. Despite this fact, analysts of a rival company have found a vulnerability in the rules which can guarantee making profit from each round of the game: get at least 0.55 of the chocolate piece in total in each round.

You have to develop such a strategy which gains at least 0.55 of the chocolate piece in total in each round, before the casino closes this vulnerability.

Interaction Protocol

In the first line, you are given two integers t and n: the number of rounds in which you must win and the number of turns that will be made in each round ($1 \le t \le 1000$; $1 \le n \le 30$; n is odd).

After that, you must win t rounds of the game. At the start of each round, the croupier has a whole chocolate bar which is labeled by 0. Each next piece of chocolate is labeled with the number of the turn when a player cuts this piece from the opponent's piece and takes it for himself. There will be n turns, the first and the last will be made by your program.

In your turn, you choose a piece of chocolate with some label i (i should be even because all the croupier's pieces have even numbers, and do not exceed the number of the current turn), which belongs to the croupier, and the percentage x ($\frac{1}{3} \le x \le \frac{2}{3}$) of this piece which you take for yourself. When you choose i and x, you should print these two numbers. The number x should be printed with at least ten digits after the decimal point (the more the better). If your x is less than $\frac{1}{3}$, it will be considered as $\frac{1}{3}$, and if it is bigger than $\frac{2}{3}$, it will be considered as $\frac{2}{3}$.

In his turn, the croupier randomly chooses one of your pieces, a uniformly distributed random odd i not exceeding the number of the current turn, takes $\frac{2}{3}$ of this piece for himself, and prints these numbers. The fraction $\frac{2}{3}$ is always printed as "0.6666666667".

After your last turn, the interactor checks if you have at least $0.55-10^{-9}$ of the initial chocolate. If not, the interactor prints 0, and your program should terminate: you don't pass the test and get "`Wrong Answer`" on this test. Otherwise, the interactor prints 1, and a new round starts immediately, where the interactor is waiting for your first turn again.

Example

standard input	standard output
2 5 1 0.6666666667 3 0.6666666667 1 1 0.6666666667 3 0.6666666667 0	0 0.5 2 0.6666666667 0 0.6666666667 0 0.6666666667 2 0.6666666667 0 0.6666666667

Note

Let's take a look at what happens in the first sample. The participant must win in two rounds.

The first round:

- Before the first turn, the participant doesn't have any pieces of chocolate, and the jury has one piece of size 1.
- After the first turn, the participant has one piece of size $\frac{1}{2}$, and the jury has one piece of size $\frac{1}{2}$.
- After the second turn, the participant has one piece of size $\frac{1}{6}$, and the jury has pieces of sizes $\frac{1}{2}$ and $\frac{1}{3}$.
- After the third turn, the participant has pieces of sizes $\frac{1}{6}$ and $\frac{2}{9}$, and the jury has pieces of sizes $\frac{1}{2}$ and $\frac{1}{9}$.
- After the fourth turn, the participant has pieces of sizes $\frac{1}{6}$ and $\frac{2}{27}$, and the jury has pieces of sizes $\frac{1}{2}$, $\frac{1}{9}$ and $\frac{4}{27}$.
- After the fifth turn, the participant has pieces of sizes $\frac{1}{6}$, $\frac{2}{27}$ and $\frac{1}{3}$, and the jury has pieces of sizes $\frac{1}{6}$, $\frac{1}{9}$ and $\frac{4}{27}$.

At the end, the participant has $\frac{1}{6} + \frac{2}{27} + \frac{1}{3} \approx 0.574$ which is bigger than 0.55: the participant wins the round.

The second round:

- Before the first turn, the participant doesn't have any pieces of chocolate, and the jury has one piece of size 1.
- After the first turn, the participant has one piece of size $\frac{2}{3}$, and the jury has one piece of size $\frac{1}{3}$.
- After the second turn, the participant has one piece of size $\frac{2}{9}$, and the jury has pieces of sizes $\frac{1}{3}$ and $\frac{4}{9}$.
- After the third turn, the participant has pieces of sizes $\frac{2}{9}$ and $\frac{8}{27}$, and the jury has pieces of sizes $\frac{1}{3}$ and $\frac{2}{27}$.
- After the fourth turn, the participant has pieces of sizes $\frac{2}{9}$ and $\frac{8}{81}$, and the jury has pieces of sizes $\frac{1}{3}$, $\frac{2}{27}$ and $\frac{16}{81}$.

- After the fifth turn, the participant has pieces of sizes $\frac{2}{9}$, $\frac{8}{81}$ and $\frac{2}{9}$, and the jury has pieces of sizes $\frac{1}{9}$, $\frac{2}{27}$ and $\frac{16}{81}$.

At the end, the participant has $\frac{2}{9} + \frac{8}{81} + \frac{2}{9} \approx 0.543$ which is less than 0.55: the participant loses the round, and the interactor prints 0, which means that solution got "`Wrong Answer`" on this test.

Problem D. DotA Quals

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

Today, instead of studying for the coming exams, a student with a nickname "Idned" has decided to take part in an open qualification for a huge DotA (Development of the Algorithms) tournament. The qualification is going to be a single-elimination tournament with 2^n participants, and Idned is one of them. There will be n rounds in total. All other remaining participants will be randomly divided into pairs for each round with equal chances for any possible division. In each pair, the participants will play against each other, and the loser will quit the tournament (and will not participate in the further rounds).

Each participant has a unique rating, and Idned's rating is k-th highest. Idned is sure that the outcome of each game is fully determined by the ratings of two participants, and whoever has a higher rating will win. Using this assumption, can you determine the expected number of rounds in which Idned will take part?

Input

The input contains two integers n and k: the total number of rounds and Idned's position in the overall rating ($1 \le n \le 10$; $1 \le k \le 2^n$).

Output

Output the expected number of rounds.

Your answer must be correct to within an absolute or relative error of 10^{-9}. Formally, let your answer be a, and the jury's answer be b. Your answer will be considered correct if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-9}$.

Examples

standard input	standard output
2 2	1.666666666667
3 5	1.457142857143

Problem E. Enumeration of Tournaments

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

Suppose that n players participate in a single-elimination tournament. The tournament goes round-by-round. Let k be the number of remaining players at the beginning of the round. Then they will randomly form $[\frac{k}{2}]$ pairs. Any possible partition can be chosen with equal probability. In each pair, two players play against each other, one of them wins, and the loser quits the tournament. The remaining $[\frac{k+1}{2}]$ advance to the next round.

It is known that each participant has a unique rating, and the outcome of each game is completely determined by the ratings: whoever has higher rating will win. So, the only thing that affects the schedule of the tournament is the random partitions into pairs each round. Can you calculate the number of different tournaments that could occur? Two tournaments are called different if there is a game (between two participants) in one of the tournaments that doesn't occur in the other tournament. As the answer can be rather large, calculate this number modulo 2^{64}.

Input

The input contains one integer n: the initial number of players in the tournament ($1 \le n \le 10^{18}$).

Output

Output the number of different tournaments modulo 2^{64}.

Examples

standard input	standard output
4	3
5	45

Problem F. Fresh Matrix

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	256 megabytes

A matrix of 0s and 1s is *good* if there are no two 1s in two matrix cells which share a side.

A matrix of 0s and 1s is *connected* if between all pairs of 0s there is a path which doesn't contain any 1s, and every two consecutive cells of the path share a side.

How many *good connected* matrices of 0s and 1s with n rows and m columns are there? As the answer can be rather big, print only its remainder modulo prime p.

Input

On the first line, you are given three integers n, m, and p: the number of rows and columns in the matrix and an integer you should use for taking the modulo ($2 \le n \le 11$; $1 \le m \le 10^9$; $2 \le p \le 10^9$; p is prime).

Output

Print one integer: the number of good connected matrices modulo p.

Examples

standard input	standard output
2 2 998244353	5
4 1 998244353	4
4 5 998244353	2749

Problem G. Game of Chairs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	256 mebibytes

You are participating in a strange game. There are n chairs in a row at a distance of one meter from each other. Each chair is painted by one of c different colors.

At the beginning, you can sit on any chair. Then the jury will choose a color and announce it. Each color has $\frac{1}{c}$ probability to be chosen. Your task is to move to any chair of this color.

Of course, you will move to the nearest appropriate chair. If you are already sitting on it, you will not move at all.

You want to pick a chair to sit at the beginning to minimize the expected distance which you will have to walk.

Input

In the first line there are two integers n and c: the number of chairs and the number of colors ($1 \le c \le n \le 10^6$).

The second line contains n integers a_i: the colors of chairs ($1 \le a_i \le c$). There is at least one chair of each color.

Output

Output the expected distance as an irreducible fraction.

Examples

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

Problem H. Hung Fu

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

The are two integer arrays a and b of length n. Consider the following formula:

$$\sum_{i=1}^{n} \min_{1 \le j \le i} a_i \oplus b_j.$$

You are practicing the calculation of the result of the above formula, and you have noticed that the order of elements in the arrays matters. Now you want to minimize the result of the calculation by permuting the elements of arrays a and b. More formally, you want to find such a permutation p that minimizes the following function:

$$F(p) = \sum_{i=1}^{n} \min_{1 \le j \le i} a_{p_i} \oplus b_{p_j}.$$

Find and output the lexicographically smallest permutation p that minimizes the function.

Input

On the first line, you are given a single integer n: the size of arrays ($1 \le n \le 50$).

On the second line, you are given n integers a_i: the elements of array a ($0 \le a_i \le 1,000,000$).

On the third line, you are given n integers b_i: the elements of array b ($0 \le b_i \le 1,000,000$).

Output

On the first line, output a single integer: the minimum possible result of the function. On the second line, output n integers: the lexicographically smallest permutation p that minimizes the result of the function.

Example

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

Problem I. Is It a p-drome?

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	256 mebibytes

Let's suppose that we have fixed a permutation p with length n. We say that a string t is a p-drome if it has the length n, and for all the characters of this string, it is true that $t_i = t_{p_i}$.

You have a string s and a permutation p. For each substring of s of length n, you have to find out if it is a p-drome or not.

Input

On the first line, you are given three integers n, m, and c: the length of the permutation, the length of the string and the size of the alphabet of the string ($1 \le n \le m \le 500\,000$; $1 \le c \le 500\,000$).

On the second line, you are given n integers p_i: the permutation itsekf ($1 \le p_i \le n$; $p_i \ne p_j$ if $i \ne j$).

On the third line, you are given m integers s_i: the initial string ($1 \le s_i \le c$).

Output

Print $m - n + 1$ characters without spaces: i-th character must be "1" if substring $s_i \dots s_{i+n-1}$ is a p-drome, and "0" otherwise.

Examples

standard input	standard output
3 5 1 1 2 3 1 1 1 1 1	111
3 7 3 3 2 1 1 2 1 3 1 2 1	10101

Problem J. Journey

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

There are n cells along a straight line numbered from 1 to n. Each cell i contains a number a_i. Initially, the player is in the cell number 1 with the number h_0 in his hand.

If the player is in a cell number p ($1 \le p \le n$) with a number h in hand, he can jump to the cell number $p + a_p$ or to the cell number $p + h$. It is forbidden to leave the field. After the jump, the new number in the player's hand is equal to the length of the last jump.

You have to calculate the number of paths from the cell 1 to the cell n. Two paths are considered different if their sets of visited cells are different. Print the answer modulo 998 244 353.

Input

The first line contains two integers n and h_0: the number of cells on the line and the number in the player's hand before the start of the path ($2 \le n \le 100\,000$; $1 \le h_0 \le n - 1$).

The second line contains n integers a_1, a_2, ..., a_n. Here, a_i is the number in i-th cell ($1 \le a_i \le n - 1$).

Output

Print a single integer: the number of different paths modulo 998 244 353.

Example

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

Problem K. Kid's Nightmare

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

Vasya likes trees and hates cycles in graphs very much. Every time when he faces a cactus problem at a contest, he gets upset, and then he starts a discussion at one forum which is well-known in the community.

Today, during the night before an important contest at the training camp, Vasya has a nightmare. In this nightmare he sees a connected undirected graph with a lot of cycles: cycles with cycles inside with cycles inside... The graph is so terrible that for every simple cycle with at least four vertices, there is an edge which connects two non-adjacent vertices of the cycle.

The only thing that Vasya can do is to delete a vertex and all the edges that are incident to this vertex. Help Vasya to delete some vertices, so that the number of remaining vertices is as large as possible, but there are no cycles at all in the remaining graph. Note that the graph may become disconnected after deletion of vertices.

Input

On the first line, you are given two integers n and m: the numbers of vertices and edges in the graph ($1 \le n, m \le 200\,000$).

Each of the next m lines contains a pair of integers u and v: the numbers of vertices that are connected by an edge ($1 \le u, v \le n$).

It is guaranteed that the graph is connected, there are no loops, no multiple edges, and for every simple cycle with at least four vertices, there is an edge which connects two non-adjacent vertices of the cycle.

Output

On the first line, print one integer t: the maximal number of vertices which can be left in the graph.

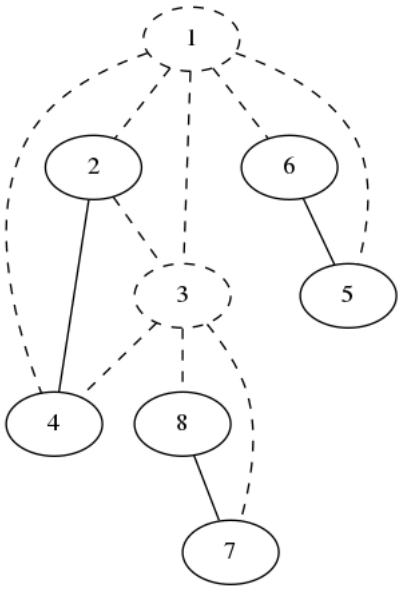
On the second line, print t integers: the numbers of vertices which can remain in the graph. If there are several solutions, any of them will be accepted.

Examples

standard input	standard output
3 2 1 2 2 3	3 1 2 3
3 3 1 2 2 3 3 1	2 1 2
8 12 1 2 1 3 1 4 2 3 2 4 3 4 1 6 6 5 1 5 3 8 3 7 8 7	6 2 4 5 6 7 8

Note

In the picture below, you can see a graph from sample test 3. Solid vertices and edges are those which are present in the resulting graph, and dashed vertices and edges are the deleted ones.



Problem L. Lazy Student

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

The Algorithms course exam is coming soon and there will be k days when you can attempt to pass it. You can come on any subset of the days until you finally pass the exam. A young lazy student Raccoon definitely wants to pass the exam, but at the same time he wants to learn the smallest possible number of topics. It means that if there is only one exam day left, Raccoon will learn all the topics. On the other hand, if there are 10 attempts, he may hope that it is enough to learn just a third of all the topics.

Let's describe the process in more detail. Let all the topics be a segment $[0, 1]$. Raccoon learns the first x topics (which is a real number from 0 to 1). When Raccoon comes to an exam-day, he will pass with probability x. Raccoon may learn any amount of topics before exams or in-between until he passes the exam. He never forgets what he already has learned. For example, Raccoon may learn $\frac{1}{3}$ before the first exam day and up-learn $\frac{1}{6}$ after a failure which will make a total of $\frac{1}{2}$ learned topics. If Raccoon fails on all the days before the last one, he will make sure that he will know all the topics before the last attempt (he definitely wants to pass the exam, remember?).

Raccoon is lazy and doesn't want to learn a lot of topics. He is going to choose a strategy that will provide him with the least expected amount of topics that he needs to learn. What is this expected amount?

Input

The input contains a single integer k: the number of attempts to pass the exam ($1 \le k \le 10^{18}$).

Output

The answer can be represented as an irreducible fraction $\frac{P}{Q}$. You are asked to output $P \bmod M$ and $Q \bmod M$ separated by a space, where $M = 1\,000\,000\,007$.

Examples

standard input	standard output
2	3 4
3	39 64

Note

For the first input (when there are just 2 exam-days) Raccoon may learn $\frac{1}{2}$ of all topics before the first attempt and another $\frac{1}{2}$ if he fails on the first day. The expected amount of topics to be learned is therefore $\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot 1 = \frac{3}{4}$.