Problem A. Binary Robots

Input file:	`binary.in`
Output file:	*`standard output`*
Time limit:	7 seconds
Memory limit:	64 mebibytes

A binary robot is a kind of robot that can perform two kinds of actions (e.g., writing and erasing, eating and thinking). However, a robot cannot perform both actions simultaneously. Some binary robots are defective and are able to perform only one type of action.

Byteasar is the head of a company producing binary robots. The company owns n robots, each of which may perform particular actions and has a particular lease price. Byteasar has received m lease offers, each of which requires a different kind of action to be performed. Each of the n robots can be assigned to at most one offer related to an action that it is able to perform. Byteasar does not need to lease all the robots or to accept all the offers. Write a program that computes the maximal profit that Byteasar can achieve.

Input

The first line of input holds three integers n, m and q ($1 \le n, m \le 1\,000\,000$, $0 \le q \le 2n$) denoting the number of robots, the number of offers (i.e., the number of considered actions) and the total number of robot–action pairs. Robots are numbered from 1 to n, whereas actions are numbered from 1 to m.

The second line of input contains n integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le 1\,000\,000\,000$) that denote the lease prices for respective robots.

The following q lines contain two integers a_i, b_i each ($1 \le a_i \le n$, $1 \le b_i \le m$). Such numbers denote that the robot number a_i is able to perform the action specified by the offer number b_i. Each pair (a_i, b_i) appears in the input at most once. For each particular $x = 1, 2, \ldots, n$, the input contains one or two pairs of the form (x, y).

Output

Your program should output exactly one integer: the maximum profit Byteasar can achieve.

Examples

binary.in	standard output
3 2 4 3 1 4 1 1 2 1 2 2 3 2	7

Problem B. Borders

Input file:	`borders.in`
Output file:	*`standard output`*
Time limit:	2 seconds
Memory limit:	64 mebibytes

A word v is called a border of the word w if v is both a prefix (an initial fragment) and a suffix (a trailing fragment) of the word w. A border of w is called proper if it is non-empty and shorter than w. Let $B(w)$ denote the number of proper borders of the word w. By $w[i, j]$ we denote a subword of w starting at the position number i and ending at the position number j. The positions are numbered starting from 1.

You are given a word w. Your task is to compute the total number of proper borders of all subwords of the word w, that is, to compute the following sum:

$$\sum_{1 \le i \le j \le |w|} B(w[i, j]).$$

Input

The first and only line of input contains the word w that consists of at least 1 and at most 10^5 lower-case letters from the English alphabet.

Output

The total number of proper borders of all subwords of the word w should be output.

Examples

borders.in	standard output
ababa	7

Problem C. Dances

Input file:	`dances.in`
Output file:	*`standard output`*
Time limit:	2 seconds
Memory limit:	64 mebibytes

The Song and Dance Ensemble of the University of Bytetown is preparing a new performance. There are n boys and $2n - 1$ girls dancing in the ensemble. For simplicity, the boys are numbered 1 through n and the girls are numbered 1 through $2n - 1$.

During the preparations for the performance, the boy number i practiced only with the girls with numbers from 1 to $2i - 1$ (for $i = 1, 2, \ldots, n$).

The final program of the performance requires forming r dancing pairs (each pair consisting of a boy and a girl). How many ways of forming such a set of pairs exist, provided that only boys and girls who practiced together can be paired?

Input

The first and only line of input contains two integers n and r ($1 \le r \le n \le 500\,000$).

Output

Your program should output the number of different pairings modulo $10^9 + 7$.

Examples

dances.in	standard output
3 2	18

Problem D. Free Calls

Input file:	`free.in`
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	64 mebibytes

A cell phone company named Bytel has prepared a super-awesome offer for its customers. Byteasar is responsible for advertising the offer. His job consists in calling the customers of the company by himself and informing them about the offer. Byteasar has found this task quite time-consuming, so he would like to optimize his job in some way.

Bytel has n customers, numbered 1 through n. Each customer can perform free unlimited calls to one number selected by the customer. Byteasar thinks that the new offer is so awesome that each customer informed about it will share her knowledge with someone else. As Bytel customers are known to optimize their costs, each customer would inform only one person about the offer, namely the one she can call for free.

Byteasar has decided to perform exactly k calls to the company's customers. He wonders which customers to choose so that the number of customers eventually informed about the offer is maximized.

Input

The first line of input contains two integers n and k ($1 \le k \le n \le 100\,000$) that denote the number of Bytel's customers and the number of calls that Byteasar is going to perform. The i-th of the following n lines contains one integer a_i ($1 \le a_i \le n$) representing the free number of the customer number i (the value $a_i = i$ corresponds to a customer who has not selected any free number yet; such a customer would not inform anyone else about the offer).

Output

The first line of output should contain one integer w equal to the number of Bytel's customers eventually informed about the offer provided that Byteasar performs his calls in an optimal way. The second line should contain k integers that represent the numbers of customers that Byteasar should call. All numbers in the second line should be different. If there are multiple correct answers, your program should select any one of them.

Examples

free.in	standard output
12 2 4 1 1 9 10 3 4 8 7 5 8 7	8 6 11

Problem E. Highway Construction Plan

Input file:	`highway.in`
Output file:	*`standard output`*
Time limit:	2 seconds
Memory limit:	64 mebibytes

Byteasar, the king of Byteland, would like to improve the road infrastructure in his country. There are n towns in Byteland, numbered 1 through n. Bytetown, the capital of Byteland, has the number 1. The towns are connected by a network of m old bidirectional roads. Each town is reachable from the capital.

The highway construction plan will consist in transforming selected old roads into modern highways. Eventually, the citizens of Byteland should be able to travel between any pair of towns using the new highways.

Byteasar would like to make the construction as cheap as possible. Additionally, if there were more than d highways leading from the capital, the citizens of the capital would protest due to excessive noise of the construction. Help Byteasar find the minimal cost of a highway construction plan that satisfies all the requirements and does not make the citizens of Bytetown unhappy.

Input

The first line of input contains three integers n, m and d ($1 \le d \le n \le 2\,000$, $0 \le m \le n(n-1)/2$) that denote the number of towns in Byteland, the number of old roads and an upper limit on the number of highways adjacent to the capital.

The following m lines contain descriptions of the old roads. Each such description consists in three integers a, b, c ($1 \le a, b \le n$, $a \ne b$, $1 \le c \le 10^9$) that represent an old road connecting towns a and b, which can be transformed into a highway at the cost c. Each pair of towns is connected by at most one road.

Output

Your program should output one integer: the minimal cost of a highway construction plan. You may assume that at least one plan satisfying all the requirements exists.

Examples

highway.in	standard output
5 6 2 1 2 2 1 3 1 1 4 2 1 5 1 4 5 10 2 3 4	16

Problem F. Jolly Jumper

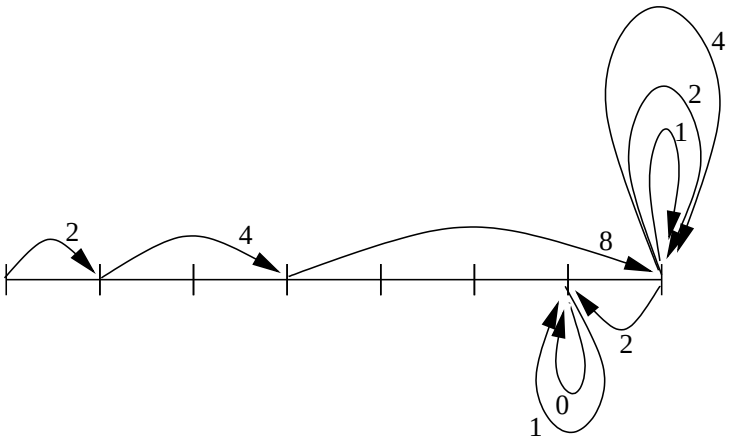
Input file:	`jumper.in`
Output file:	*`standard output`*
Time limit:	2 seconds
Memory limit:	64 mebibytes

A company is developing a special vehicle called Jolly Jumper that can traverse huge distances very quickly. The vehicle moves by jumping, and thanks to its revolutionary *Double Up*^{TM} engine each successive jump can be twice as long as the previous one. Jolly Jumper is being tested on an infinitely long straight narrow road. It can only jump along the road, in either direction.

Let us explain precisely the way Jolly Jumper works. In the beginning, when the vehicle is not moving and the *Double Up*^{TM} engine is first started, the engine goes into energy state 1. When the engine is in energy state i there are two possible options:

- jump exactly i units backward or forward and switch to energy state $2i$ (a heating-up jump),
- jump directly up (hence not moving along the road) and switch to energy state $i/2$ (a cooling-down jump).

If the energy state drops below 1 the engine shuts down. The engine cannot be shut down if the energy state is 1 or more.



Sample travel of Jolly Jumper, starting at the leftmost point in energy state 1. The numbers above arrows indicate the energy state achieved *after* the jump (0 means the engine was shut down).

During tests Jolly Jumper travels between two distinct points A and B located n units apart on the infinitely long road. The technicians want to know the minimum number of jumps required to travel from A to B and shut down the engine. We assume that at the beginning of the test the engine is off (starting it up is not considered a jump).

Input

The first line of the standard input contains one integer n ($1 \le n \le 10^{100}$) — the distance between A and B.

Output

The only line of the standard output should contain one integer k — the minimum number of jumps needed to travel from A to B and shut down the engine.

Examples

jumper.in	standard output
6	9

Problem G. Sandwiches

Input file:	`sandwiches.in`
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	64 mebibytes

Bytie was trying to master the art of cooking. He was not really successful, though, so now he decided to learn how to make tasty sandwiches instead.

Bytie is going to use k ingredients in each sandwich. He has exactly k types of each ingredient (k types of cheese, k types of bread, k kinds of fruit preserves etc). Bytie has assigned to each type of an ingredient a coefficient of tastiness. In the following k days he is willing to make k different sandwiches. He will need to eat all of them, so he would like them to be as tasty as possible. The tastiness of a sandwich is the sum of tastiness coefficients of all its ingredients. Help Bytie to find out how testy will his sandwiches be.

Input

The first line of input contains one positive integer t ($1 \le t \le 10^5$) that denotes the number of test cases, described in the following lines.

The first line of a description of a test case contains one integer k ($1 \le k \le 1\,500$). The following k lines contain k non-negative integers each, with no number exceeding 10^6. The numbers in the i-th line denote the tastiness coefficients of different types of the i-th ingredient.

You can assume that the input will contain at most $2\,500\,000$ numbers in total.

Output

Your program should output k lines with the answers to the respective test cases. The answer to one test case consists in k positive integers listed in a non-increasing order that denote the tastiness of sandwiches prepared by Bytie during the k days.

Examples

sandwiches.in	standard output
2 3 1 8 5 9 2 5 10 7 6 2 1 1 1 2	27 24 24 3 3

Problem H. Number Transformations

Input file:	`transformations.in`
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	64 mebibytes

For a given positive integer n (written in decimal), we consider the following three types of transformations:

$A(n)$: extend the number n with a trailing 0

$B(n)$: extend the number n with a trailing 4

$C(n)$: divide n by 2, provided that n is even.

For example, $A(100) = 1000$, $B(100) = 1004$, $C(100) = 50$.

We start with the number 4 and perform a sequence of transformations of the type A, B and/or C. We ask whether it is possible for the sequence to end with a given positive integer n?

Input

The first line of input contains one integer k ($1 \le k \le 100\,000$) that denotes the number of test cases. Each of the following k lines contains one positive integer n_i ($1 \le n_i \le 10^{18}$, $n_i \ne 4$) that is the goal of the sequence of transformations.

Output

Your program should output exactly k lines. The i-th line should contain either a single number -1 if the requested sequence of transformations does not exist, or one non-empty word composed of letters `A`, `B` and/or `C` that represent the names of the consecutive transformations that should be performed on the number 4 to obtain the number n_i.

For each test case used for grading, if the answer is positive then the number n_i can be obtained using at most 200 transformations. The length of the sequence of transformations returned by your program may not exceed 200. If there is more than one solution, your program may output any one of them.

Examples

transformations.in	standard output
1 15	CBCCCAC

Problem I. Uniform Flow

Input file:	`uniform.in`
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	64 mebibytes

Imagine a system of n junctions, numbered from 1 to n, connected with undirected pipes in which water flows. The junction number one is the source, the n-th one is the sink. Every pipe connects exactly two distinct junctions and is assigned one number — its size, that is, the maximum velocity of water flowing in it. The water flow must adhere to the regular conservation principle: for every junction, except source and sink, the amount of water flowing into the junction is equal to the amount of water flowing out of it.

Now we add one more limitation: for each pair (J_1, J_2) of junctions the sum of water velocities over all the pipes on an arbitrary path from J_1 to J_2 is constant (for this pair of junctions). We calculate the sums in such a way, that if the water flows against direction of the path from J_1 to J_2, its velocity is negated.

Your job is to calculate the maximal flow in the network.

Input

The first line of the standard input contains one natural number n — the number of junctions in the network ($2 \le n \le 100$). The second line contains one natural number m ($1 \le m \le 5\,000$) — the number of pipes in the network. The next m lines contain the descriptions of the pipes. Each pipe is represented by three integers a_i, b_i, c_i. a_i and b_i are the numbers of junctions connected by the pipe and c_i ($0 \le c_i <= 10\,000$) is its size. There can be multiple pipes between a pair of junctions.

Output

The first line of standard output should contain exactly one floating point value — the maximal achievable flow. The number must match the exact result with the maximal absolute difference of 10^{-3}.

Examples

uniform.in	standard output
4 6 1 3 2 1 2 3 1 2 2 2 4 5 2 3 2 3 4 5	5.200000

Problem J. Deadly Waste

Input file:	`waste.in`
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	64 mebibytes

A train with an enormous quantity of deadly waste has just arrived to Bytetown. The waste should be recycled in advanced recycling facilities. In Byteland there are n towns, numbered from 1 to n, with several railway tracks connecting them. In some towns there are facilities that are able to recycle selected types of waste. There are 26 kinds of waste, denoted by small letters of the English alphabet.

Your task is to create a cheapest recycling plan for the waste. Initially the train consists of l wagons, each containing a particular kind of waste. The cost of moving a train between a pair of towns is proportional to the distance between the towns and does not depend on the length of the train. In any town a train can be divided into two parts: several consecutive wagons may be disconnected from the end of the train and attached to a separate locomotive. This operation can be performed many times, however no trains can be connected back to form a single train. Eventually each train must end its track in a town that contains a facility able to recycle all the waste from this train.

Input

The input contains several test cases.

The first line of a test case contains four integers n, m, l and s ($1 \le n \le 1\,000$, $0 \le m \le 10\,000$, $1 \le l \le 40$, $1 \le s \le n$) that denote the number of towns, the number of railway tracks, the initial length of the train and the number of the town corresponding to Bytetown, respectively.

The following n lines provide a description of the towns. The i-th of those lines contains an integer k_i ($0 \le k_i \le 26$) and a word composed of k_i letters denoting the kinds of waste that can be recycled in the i-th town.

The following m lines describe the railway connections. The i-th of these lines contains three integers a_i, b_i and w_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le w_i \le 1\,000$) that represent a bidirectional railway track of length w_i connecting the towns number a_i and b_i.

The last line contains a word of length l that describes the kinds of waste in the consecutive wagons of the train.

The input ends with a test case with $n = m = l = s = 0$ that should not be processed. The total size of input does not exceed 1.5 MB.

Output

For each test case one line should be output. This line should contain a single integer denoting the minimal cost of transporting all the waste, or -1 if a requested recycling plan does not exist.

Examples

waste.in	standard output
3 3 4 1 1 a 0 1 b 1 3 4 3 2 1 2 1 2 abab 0 0 0 0	6