Problem A. Bitwise-Xor to Zeroes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	512 mebibytes

You are given an array of length n, which consists of non-negative natural numbers. In one move you can choose a pair of indices i, j ($1 \le i, j \le n$, possibly $i = j$) and perform this pseudocode:
$x = a_i \oplus a_j$
$a_i = x$
$a_j = x$

What is the minimum possible number of moves needed to make the whole array containing only zeroes? But don't be so fast, answering once for whole array would be too easy. There are many queries, in each of them you should assume that there exists only some interval of input array, and your goal is to make this interval containing only zeroes. Queries are independent – in each query you start from an interval of the initial array. It can be proven that it's always possible to turn an interval into zeroes.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 3 \cdot 10^5$) – the length of the array and number of queries. The second line contains n integers a_i ($0 \le a_i \le 10^9$) – the array. Each of the next m lines contains two integers a and b ($1 \le a \le b \le n$) describing a query about interval $[a, b]$.

Output

Output m lines, where i-th of them should contain answer for i-th query – minimal number of moves to make i-th interval containing only zeroes.

Examples

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

Note

In third query we can choose two elements with value 3 and replace them both with value 0. Next, we can choose last index as both i and j and turn its value to 0. Interval will change as follows:
3 3 5 → 0 0 5 → 0 0 0.

Problem E. Exact Covering

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	512 mebibytes

You are given a tree with n vertices and with weighted edges. Simple path is a sequence of edges from the tree, such that every two consecutive edges from the sequence have common vertex, to which both of them are incident, no edge appears more than once in this sequence and no vertex is adjacent to more than two edges from the path. Your task is to cover the tree with minimal number of simple paths (choose some multiset of simple paths) such that every edge lies on exactly as many paths as its weight. To make task more difficult, weights are still changing and after every change you have to find the minimum number of paths needed to cover all edges.

Input

The first line of the input contains one integer n ($1 \le n \le 5 \cdot 10^5$) – the number of vertices in tree. Each of the next $n-1$ lines contains three integers a, b and c ($1 \le a, b \le n;\ 0 \le c \le 10^9$) describing an edge between vertices a and b with weight c. Next line contains one integer q ($1 \le q \le 5 \cdot 10^5$) – number of changes. Each of the next q lines contains three integers a, b and c ($1 \le a, b \le n;\ 0 \le c \le 10^9$) which mean that edge between vertices a and b from now will have weight equal to c.

Output

Output $q+1$ lines. i-th line should contain one integer x, being the minimal possible number of paths needed to cover the tree after $i-1$ changes. Note that first number in the output should be equal for answer before any changes.

Example

standard input	standard output
4 1 2 2 2 3 2 2 4 1 2 2 4 2 2 1 100	3 3 100

Problem F. Foreheads Game

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

Have you ever heard about **Foreheads game**? This game is played by k players. In this game players use only one suit from typical deck of cards (so there are 13 cards). At the beginning all cards are shuffled, everybody picks one card and puts it on his/her forehead, so everybody knows card of every other player, but doesn't know his own. Next, players one by one are guessing the relative rank of their card. For example if first player has a 5 and two other players have a 9 and an ace, then he has the third highest card. After guessing, players are in some way rewarded with points based on how close they were.

In this task we'll look on a little bit modified version of this game. k players are sitting around a table and every of them has a card on his/her forehead. Now, players cyclically are telling if they know their rank. Cyclically means that for example for $k = 3$ they are telling it in order $1, 2, 3, 1, 2, 3, 1, 2, 3\ldots$ if we number them with numbers from 1 to k. If a current player knows his/her rank the game ends instantly. Otherwise next player's turn starts.

Note that there is no guessing in this version of the game, i.e. we are assuming that if player isn't sure about his/her rank, then he/she won't lie and if he can deduce his position, then he will for sure do it, and end the game. In other words, every player is infinitely smart. Your task is to calculate in which turn somebody will tell that he knows his/her rank or tell that game will go on infinitely.

Input

The first line of the input contains two integers k and t ($2 \le k \le 5$; $1 \le t \le 30$) – number of players and number of games they play. Games are independent from each other. Each of the next t lines describes one game and contains k different space-separated characters from set $\{2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A\}$, where i-th of them represents i-th player's card.

Output

Print t lines. In i-th of them there should be one one integer, which means in which turn in i-th game somebody will tell that he knows his rank. If i-th game won't end, print -1.

Examples

standard input	standard output
3 1 K 9 2	2
5 2 4 8 Q 9 7 7 A K 2 3	13 1

Note

In first sample second player can see that third player has a 2, so he deduces that if he'd have a 3 or an ace, then first player would know his rank. First player doesn't know his rank, so second player deduces that he has second highest card and ends the game in second turn.

Problem H. Heavier Coins

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

There are n coins. Each of them has a weight which can be represented with some strictly positive real number. Unfortunately you don't know much about exact weights – for every two different coins first one can be lighter than second one, heavier than second one, or even have exactly this same weight. You only know about results of m weightings. In i-th of them a_i-th and b_i-th coin were compared and it turned out that the first one is strictly heavier than the second one.

Now you want to choose sequence of non-empty subsets of coins, such that for every two different subsets from this sequence the one which appears in this sequence later is for sure strictly heavier than the one which appears earlier. Weight of subset of coins is simply sum of weights of its coins. By "for sure" we mean that there cannot exist any set of coins' weights which satisfies weightings and for which later subset isn't heavier than earlier subset. Your task is to find maximum possible number of subsets in this sequence and to print this number modulo $10^9 + 7$.

Input

The first line of the input contains two integers n and m ($1 \le n \le 10^6$; $0 \le m \le 10^6$) – number of coins and number of weightings. Each of the next m lines contain two distinct numbers a_i and b_i ($1 \le a_i, b_i \le n$) which mean that a_i-th coin is strictly heavier than b_i-th coin. It's guaranteed that these weightings don't contradict with themselves, i.e. there exists at least one set of coins' weights which satisfies these weightings.

Output

Output one integer which is equal to maximum possible number of subsets in sequence taken modulo $10^9 + 7$.

Examples

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

Note

In first test we can choose following sequence of subsets: $\{2\}, \{1\}, \{1, 2\}, \{1, 2, 3\}$. Note that it's forbidden to have in sequence both subsets $\{1\}$ and $\{2, 3\}$, because it's not clear which subset is heavier.

Problem K. KMR

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

KMR is an ACM ICPC team, whose members are dreaming about achieving a bronze medal at the finals. They've found in the internet new training course which can be described as DAG (directed acyclic graph). Vertices are states in training and edges are tasks to solve, vertices are numbered from 1 to n. Every programmer at the beginning of course is in vertex 1 and in any moment he can choose some edge which is outgoing from his vertex and solve problem assigned to it, which will effect in changing actual vertex of this programmer and moving him to end of this edge. Course ends when programmer is in vertex n. Tasks are pairwise different. All three members of KMR want to complete this course independently, but they know that it makes no sense if there is a task solved by all three of them, so they don't want to allow such situation to take place. They are wondering what is the number of ways to complete the course by their team. Your task is to calculate this number and to print it modulo $10^9 + 7$.

Two ways of completing the course by team are considered different if there exists a member and a task such that this member solved this task in the first way, but didn't solve it in the second way.

Input

The first line of the input contains two integers n and m ($2 \le n \le 2000$; $1 \le m \le 10000$) – the number of vertices and the number of edges (tasks). Each of the next m lines contains two integers a and b ($1 \le a, b \le n$) which denote an edge from vertex a to vertex b (representing a task which if solved will move programmer from vertex a to vertex b).

It's guaranteed that described graph is a DAG. It's also guaranteed that by starting in vertex 1 it's possible to achieve every other vertex and by starting in any vertex it's possible to achieve vertex n.

Output

Output one integer which stands for number of different ways of completing the course by KMR taken modulo $10^9 + 7$.

Example

standard input	standard output
4 5 1 2 2 3 3 4 1 3 2 4	12
3 4 1 2 1 2 2 3 2 3	36

Problem L. Level of Acidity

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

You are gathering readings of acidity level in a very long river in order to determine the health of the river. You have placed N sensors in the river, and each sensor gives an integer reading R. For the purposes of your research, you would like to know the frequency of each reading, and find the absolute difference between the two most frequent readings.

If there are more than two readings that have the highest frequency, the difference computed should be the largest such absolute difference between two readings with this frequency. If there is only one reading with the largest frequency, but more than one reading with the second largest fre- quency, the difference computed should be the largest absolute difference between the most fre- quently occurring reading and any of the readings which occur with second-highest frequency.

Input

The first line of input will be the integer N ($2 \le N \le 2 \cdot 10^6$), the number of sensors. The next N lines each contain the reading for that sensor, which is an integer R ($1 \le R \le 1000$). You should assume that there are at least two different readings in the input.

Output

Output the positive integer value representing the absolute difference between the two most fre- quently occurring readings, subject to the tie-breaking rules outlined above.

Examples

standard input	standard output
5 1 1 1 4 3	3
4 10 6 1 8	9

Problem M. Matches

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

Consider next game with matches.

Let E be a set consisted of elements {/, \, -, |, .}, i. e., the elements of this set are: one match as the diagonal of a unit square, one match as the segment connecting the middle points of the opposite sides (horizontal and vertical) of unit square, and the empty space, respectively.

Consider a rectangle $n \times m$ consisting of unit squares, each of which is labeled by exactly one of the elements of the set E. Let us call «a segment in the rectangle» a set of unit squares in the specified rectangle such that the labels on elements of the set form a continuous segment, and this set can not be extended to any other set, labels on the elements of which also form a continuous segment.

Your are given such a rectangle. Find the total number of the segments in this rectangle.

Input

First line of input contains two integers n and m ($1 \le n, m \le 100$) — height and width of the box, respectively.

Each of next n lines contain m characters from set «/», «\», «-», «|», «.». i-th character in j-th of those lines denotes the label on unit square (i, j). You may assume that atleast one of those characters does not equals '.'.

Output

Print one integer — number of segments in the given rectangle.

Examples

standard input	standard output
4 5 ..|/. ..|.. ./|.. -----	4

Problem N. New Soccer

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

A «new soccer» game operates under slightly different soccer rules. Each team consists of 99 players. A goal is only counted if the 4 players, in order, who touched the ball prior to the goal have jersey numbers that are in strictly increasing numeric order with the highest number being the goal scorer. Players have jerseys numbered from 1 to 99 (and each jersey number is worn by exactly one player). Given a jersey number of the goal scorer, indicate how many possible combinations of players can produce a valid goal.

Input

The input will be the positive integer J ($1 \leq J \leq 99$), which is the jersey number of the goal scorer.

Output

The output will be one line containing the number of possible scoring combinations that could have J as the goal scoring jersey number.

Examples

`standard inputstandard output`	`2 seconds`
4	1
2	0
90	113564

Problem O. Ordered Subsets

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

Probability and statistics have numerous examples where people want to analyze how to select m items from a set of n items. For example, consider the following 5 items: 0, 1, 2, 3, and 4. Suppose you were interested in selecting 3 items from this list. If we were to list all the ways to select 3 items from this set, we could list these 10 possibilities: 012, 013, 014, 023, 024, 034, 123, 124, 134, 234.

There are many ways to order the elements of this set, but for this problem we are interested in listing the combinations lexicographically. That is, if we view the above list as 3-letter "words", 012 comes before 013 and 134. Note that each 3-letter combination is itself written lexicographically, as is the list of all 3-item combinations. That is, the digits of the item 014 are written in increasing order (not 041, for example).

The problem you are asked to solve is, given that you wish to consider all the ways to select M elements from a set of N elements, identify the i-th possibility, if all of those possibilities were listed lexicographically.

In this problem N can be as large as 36. So, we will use the uppercase letters to fill out the set of objects, where the ordering is: 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ. That is, the lexicographical ordering will use these 36 characters in determining "alphabetical" order. ('0' comes before 'A', for example).

Input

Each line of the input will represent one test case and will contain 3 integers: N, M, and I. N represents the total number of objects to select from, M represent the number of objects to select, and I represents the position of the ith combination, where the combinations are ordered as described above. N will be at least 1 and at most 36. M will be at least 1 and no more than N. I will be a legal value for M and N (i.e., at least 1 and at most $N!/(M!(N-M)!)$. The final line of input will be when N is equal to 0. Do not produce any output for this last line of input. There will be at most 2000 problems to solve in the input file.

Output

For each input, display the ith combination using the format shown below, with the problem number, followed by a colon and space, followed by the i-th combination.

Example

standard input	standard output
5 3 1 5 3 2 5 3 10 4 2 1 4 2 6 10 5 1 10 5 252 11 11 1 36 2 1 36 2 2 36 2 630 36 36 1 36 18 1 36 18 9075135300 0 0 0	1: 012 2: 013 3: 234 4: 01 5: 23 6: 01234 7: 56789 8: 0123456789A 9: 01 10: 02 11: YZ 12: 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ 13: 0123456789ABCDEFGH 14: IJKLMNOPQRSTUVWXYZ

Problem P. Process

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

Consider the following number operation (let's call it the root-rounding operation) that begins with some positive integer, and ends with a positive integer:

1. Begin with a positive integer, n.
2. Compute the largest integer that is less than or equal to the square root of n. Call this result m.
3. If $n + m$ is even, then the final result is $n + m$. Otherwise, the final result is $n - m$.

One might consider next process: taking a number, computing its root-rounded result, and then taking that result and computing its root-rounded result, and so on. For example, begin with 25, and the result is 30. Then, take 30 and compute its result: 25. 25 gives 30. 30 gives 25, and so on. This is referred to as a "2-cycle". On the other hand, if you begin with 24, the resulting sequence is a 13-cycle: 24, 28, 23, 19, 15, 18, 22, 26, 21, 17, 13, 16, 20, 24.

Notice, that the last number creates a cycle. We refer to this particular cycle as a 13-cycle, because it contains exactly 13 non-repeating integers, with the 14th integer simply repeating the first integer.

Input

First line of input file contains integer T ($1 \le T \le 200$) — number of test cases. Each test case is given on separate line and contains one positive integer, with value at most 10^6. You will determine the length of each integer's root-rounding cycle.

Output

Formatting your results as shown in the sample output below (the input, followed by a colon and space, followed by the length of its root-rounding cycle).

Example

standard input	standard output
3 25 24 1	25: 2-cycle 24: 13-cycle 1: 2-cycle

Problem Q. Quasiroman Numbers

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

This question involves calculating the value of «quasiroman» numbers which are a combination of Arabic digits and Roman numerals.

An quasiroman number is of the form $A_1R_1A_2R_2 \dots A_nR_n$, where each A_i is an Arabic digit, and each R_i is a Roman numeral. Each pair A_iR_i contributes a value described below, and by adding or subtracting these values together we get the value of the entire quasiroman number. An Arabic digit A can be 0, 1, 2, 3, 4, 5, 6, 7, 8 or 9. A Roman numeral R is one of the seven letters I, V, X, L, C, D, or M. Each Roman numeral has a base value: 1, 5, 10, 50, 100, 500 or 1000, respectively.

The value of a pair AR is A times the base value of R. Normally, you add up the values of the pairs to get the overall value. However, wherever there are consecutive symbols $ARA'R'$ with R' having a strictly bigger base value than R, the value of pair AR must be subtracted from the total, instead of being added.

For example, the number $3M1D2C$ has the value $3 \cdot 1000 + 1 \cdot 500 + 2 \cdot 100 = 3700$ and $3X2I4X$ has the value $3 \cdot 10 - 2 \cdot 1 + 4 \cdot 10 = 68$.

Write a program that computes the values of quasiroman numbers.

Input

The input is a valid quasiroman number consisting of between 2 and 20 symbols.

Output

The output is the decimal value of the given quasiroman number.

Examples

standard input	standard output
3M1D2C	3700
2I3I2X9V1X	-16