Problem A. Three Arrays

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

You are given three arrays: a containing n_a elements, b containing n_b elements and c containing n_c elements. These arrays are sorted in non-decreasing order: that is, for every i such that $1 \le i < n_a$ we have $a_i \le a_{i+1}$, for every j such that $1 \le j < n_b$ we have $b_j \le b_{j+1}$, and for every k such that $1 \le k < n_c$ we have $c_k \le c_{k+1}$.

Your task is to calculate the number of triples (i, j, k) such that $|a_i - b_j| \le d$, $|a_i - c_k| \le d$, and $|b_j - c_k| \le d$.

Input

The input contains one or more test cases. Each test case consists of four lines.

The first line of each test case contains four integers: d, n_a, n_b, and n_c ($1 \le d \le 10^9$, $1 \le n_a, n_b, n_c \le 5 \cdot 10^5$).

The second line contains n_a integers $a_1, a_2, \ldots, a_{n_a}$: the array a ($-10^9 \le a_i \le 10^9$).

The third line contains n_b integers $b_1, b_2, \ldots, b_{n_b}$: the array b ($-10^9 \le b_i \le 10^9$).

The fourth line contains n_c integers $c_1, c_2, \ldots, c_{n_c}$: the array c ($-10^9 \le c_i \le 10^9$).

All arrays are sorted in non-decreasing order. The total sum of n_a over all testcases does not exceed $5 \cdot 10^5$. The total sum of n_b over all testcases does not exceed $5 \cdot 10^5$. The total sum of all n_c over all testcases does not exceed $5 \cdot 10^5$. The test cases just follow one another without any special separators.

Output

For each test case, print a single integer: the number of triples (i, j, k) such that $|a_i - b_j| \le d$, $|a_i - c_k| \le d$, and $|b_j - c_k| \le d$.

Example

standard input	standard output
1 3 3 3 1 2 3 1 2 3 1 2 3 1 6 6 6 1 1 2 2 3 3 2 2 3 3 4 4 3 3 4 4 5 5	15 56

Problem C. Cover the Paths

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

You are given an undirected unweighted tree consisting of n vertices labeled by integers 1, 2, ..., n. A total of m simple paths are chosen in this tree. Each path is described as a pair of its endpoints (a_i, b_i).

Let V be the set of all vertices of the tree. We say that subset S of V is *good* if for every i such that $1 \le i \le m$, the simple path from a_i to b_i contains at least one vertex from S. We say that subset T be *the best* subset if T is a *good* subset and there is no *good* subset X such that $|X| < |T|$.

You have to find *the best* subset of V.

Input

The first line contains an integer n, the number of vertices in the tree ($1 \le n \le 10^5$).

Each of the next $n - 1$ lines describes an edge of the tree. Edge i is denoted by two integers u_i and v_i, the labels of vertices it connects ($1 \le u_i, v_i \le n$, $u_i \ne v_i$). It is guaranteed that the given edges form a tree.

The next line contains an integer m, the number of paths ($1 \le m \le 10^5$).

Each of the next m lines describes a path in the tree. Path i is denoted by two integers a_i and b_i, the labels of the endpoints ($1 \le a_i, b_i \le n$). **For some paths, it may be that $a_i = b_i$.** It is **not** guaranteed that all paths are pairwise distinct.

Output

On the first line, print the size of *the best* subset of V. On the second line, print the labels of vertices belonging to *the best* subset of V in any order.

If there are several possible solutions, print any one of them.

Examples

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

Problem D. Elevator

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

You have a very important job: you are responsible for an elevator in the new skyscraper.

There are n persons who will come to the underground parking located on floor 0 and wait for an elevator to bring them to some upper floor. Formally, i-th person comes to the elevator at moment t_i and wants to reach floor a_i. The elevator has **infinite** capacity; that is, there is no limit on the number of people using the elevator at any moment. All numbers t_i are distinct. Passengers always enter the elevator as long as it is at floor 0.

The elevator uses the following algorithm: it stays open on floor 0 until you send it to deliver passengers, then it moves to the highest floor it needs (the maximum a_i among all passengers who are currently in the elevator), distributing the passengers in the process, and returns to the parking. The elevator spends 1 unit of time to move to the next floor (or to the previous floor). The time spent for opening and closing the doors of the elevator, as well as for the passengers entering and leaving the elevator, is negligible. At moment 0, the elevator is at floor 0.

You want to minimize the moment of time when the elevator will return to floor 0 after delivering everyone.

Input

The input contains one or more test cases.

The first line of each test case contains one integer n: the number of passengers ($1 \le n \le 2 \cdot 10^5$).

Each of the following n lines contains two space-separated integers t_i and a_i: the moment of time when i-th passenger comes to the elevator, and the destination floor of i-th passenger ($1 \le t_i, a_i \le 10^9$).

All t_i in one test case are distinct, passengers appear in input in ascending order of t_i.

The sum of the values of n over all test cases does not exceed $2 \cdot 10^5$. The test cases just follow one another without any special separators.

Output

For each test case, print one integer: the minimum possible moment of time when the elevator will return after delivering all passengers.

Example

standard input	standard output
3 1 9 2 6 15 6 3 1 9 2 6 15 8	31 33

Problem G. Berland Post

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

Berland Post is the national postal service of Berland. There is exactly one post office in each of n Berland cities. Cities and their respective offices are numbered by integers from 1 to n.

There are m pairs of cities (a, b) such that there is direct post traffic from a to b. For each such pair of cities, the delivery time is known: formally, you are given m triples (a_j, b_j, d_j) meaning that each day, the post office in a_j has to send correspondence to the post office in b_j, and d_j is the time elapsed between sending the correspondence from a_j and receiving it at b_j.

Each day, all offices must be open for the equal consecutive amount of time, which is denoted as T. But opening times may differ. If the opening time of i-th office is o_i, then the closing time is $o_i + T$.

Some values of o_i are known and fixed, but some of them are up to you. Your goal is to find such values $T \geq 0$ and o_i that each office receives all the correspondence no later than at closing time, and T is the minimum possible. It is allowed for an office to receive the correspondence even before opening. Assume that each office sends the correspondence instantly after opening.

Formally, find the minimum possible non-negative T and values o_i such that $o_{a_j} + d_j \leq o_{b_j} + T$ for each of the m given triples (a_j, b_j, d_j).

Input

The input contains one or more test cases.

Each test case starts with a line containing two integers: n, the number of cities, and m, the number of direct traffic paths ($1 \leq n \leq 1000, 0 \leq m \leq 2000$).

The second line of each test case contains n tokens o_i, where o_i is either a question mark ("?") if the opening time of the office i is not given and your task is to define it, or an integer ($-10^5 \leq o_i \leq 10^5$) if the opening time of the office i is known and you can not change it.

The following m lines contain descriptions of direct traffic paths, one per line. Each line contains three integers: a_j, b_j, and d_j, denoting direct post traffic from the city a_j to the city b_j with delivery time d_j ($1 \leq a_j, b_j \leq n$, $a_j \neq b_j$, $1 \leq d_j \leq 100$). It is guaranteed that, for each pair of the cities (a, b), there is at most one direct traffic path from a to b.

The sum of all values n in a test case does not exceed 1000. The sum of all values m in a test case does not exceed 2000. The test cases just follow one another without any special separators.

Output

For each test case, print exactly two lines.

Print the minimum possible non-negative real value of T on the first line and the values o_1, o_2, ..., o_n on the second line. The values of o_i must be in the range $[-10^9, 10^9]$. Print T and o_i with absolute error of at most 10^{-4}.

The values $o_i \neq$ "?" in the input must not change. For each of the m given triples (a_j, b_j, d_j), it must be true that $o_{a_j} + d_j \leq o_{b_j} + T$.

Examples

standard input	standard output
2 1 5 7 1 2 3	1 5 7
2 2 ? ? 1 2 3 2 1 1 3 0 ? ? 3	2 9 10 0 1 -1 3

Problem J. Subsequence Sum Queries

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

You have an array a containing n integers and an integer m. You also have q queries to answer. The i-th query is described as a pair of integers (l_i, r_i). Your task is to calculate the number of such subsequences $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$ that $l_i \le j_1 < j_2 < \ldots < j_k \le r_i$ and $(a_{j_1} + a_{j_2} + \ldots + a_{j_k}) \bmod m = 0$. In other words, you need to calculate the number of subsequences of subarray $[a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}]$ such that the sum of elements in each subsequence is divisible by m.

Input

The first line contains two integers n and m: the number of elements in a and the modulo ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 20$).

The second line contains n integers a_i: the elements of array a ($0 \le a_i \le 10^9$).

The third line contains one integer q: the number of queries ($1 \le q \le 2 \cdot 10^5$).

Then q lines follow. The i-th of these lines contains two integers l_i and r_i that describe the i-th query ($1 \le l_i \le r_i \le n$).

Output

Print q lines. The i-th of them must contain the answer for the i-th query. Queries are indexed in the order they are given in the input. Since the answers can be very large, print them modulo $10^9 + 7$.

Example

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

Problem K. Consistent Occurrences

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	256 mebibytes

Let us define a *consistent set of occurrences of string t in string s* as a set of occurrences of t in s such that no two occurrences intersect (in other words, no character position in s belongs to two different occurrences).

You are given a string s consisting of n lowercase English letters, and m queries. Each query contains a single string t_i.

For each query, print the maximum size of a consistent set of occurrences of t in s.

Input

The first line contains two space-separated integers n and m: the length of string s and the number of queries ($1 \le n \le 10^5$, $1 \le m \le 10^5$).

The second line contains the string s consisting of n lowercase English letters.

Each of the next m lines contains a single string t_i consisting of lowercase English letters: the i-th query ($1 \le |t_i| \le n$, where $|t_i|$ is the length of the string t_i).

It is guaranteed that the total length of all t_i does not exceed 10^5 characters.

Output

For each query i, print one integer on a separate line: the maximum size of a consistent set of occurrences of t_i in s.

Example

standard input	standard output
6 4 aaaaaa a aa aaa aaaa	6 3 2 1

Problem M. Rectangle

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

Calculate maximum area of rectangle with diagonal of given even integer length N.

Input

Input consists of one integer N ($1 \le N \le 10^4$, $N = 2K$ for some integer K).

Output

Print maximum possible area of the rectanle with diagonal N.

Example

standard input	standard output
6	18
2	2

Problem N. HTML Copypaste

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

You are part of a team implementing an HTML editor and have been tasked with the problem of implementing the cut/copy/paste functions. One of your goals is to preserve the formatting of selected text, even though that formatting may be determined by HTML tags outside the range of the actual selected text.

You will be provided with a block of formatted text, a starting position B, and an ending position E. Your program should output the text of the substring of that text from B (**inclusive**) to E (**exclusive**), prepending and appending formatting tags as necessary so that the output is well formed and has the same format as it had in its original position.

For the purposes of this problem, format tags consist of an opening tag (such as "`<b>`"), followed by some text, followed by a closing tag (such as "`</b>`"). Opening and closing tags are paired ("`</mytag>`" closes "`<mytag>`") and are considered opened between the opening tag and the closing tag. A tag may not be closed unless it is the most recent unclosed tag (e.g., "`<i>abc<b>def</i>ghi</b>`" is illegal). A tag may not be opened if it is already open (e.g., "`<i><i>recursive i</i></i>`" is illegal).

Input

Input data will consist of 50 or less test cases. Each test case will consist of one line of input of the form B E $TEXT$ where B is an integer giving the (**inclusive**) beginning location of the substring, E is an integer giving the (**exclusive**) ending location of the substring, and $TEXT$ contains the text from which to extract the substring. The $TEXT$ begins after a single blank character immediately following E, and continues to the end of the line. B and E will be specified so that $0 \le B \le E \le length(TEXT)$.

End of input will be signaled by the line containing two -1's.

No input line will be longer than 200 characters.

The $TEXT$ will be composed of characters with an ASCII value between 32 and 126 inclusive and cannot start from a space or end by a space. Opening tags will be of the form "`<X>`" where X contains at least 1 character and is composed entirely of the characters '`a`' to '`z`', '`A`' to '`Z`, '`0`' to '`9`', and '`-`'.

Closing tags will be of the form "`</X>`". The character '`<`' will only occur in the input as the first character of an opening or closing tag.

The input text will be well formed: all opening tags will be matched with a closing tag, all closing tags will match an opening tag, each closing tag will close the most recent unclosed tag, and tags will not be recursive (each tag must be closed prior to reopening).

Neither B nor E will reference a character that is part of an opening or closing tag except for the character '`<`'.

Output

For each test case your program should print a single line containing the substring of $TEXT$ from B (inclusive) to E (exclusive), prepending the substring with opening tags and appending the substring with closing tags as necessary so that the output line is well formed and has the same set of open tags as when it was included in the original $TEXT$.

Example

standard input
0 12 Icpc! 18 23 <big>100, <bigger>1000, <biggest>10000</biggest></bigger></big> 4 4 123 0 14 :-/:-> :-) :-<-> </-> -1 -1
standard output
Icpc! <big><bigger>1000,</bigger></big> :-/ :-> :-) :-

Problem O. Cybersecurity

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

Cybersecurity is a tricky thing. Users prefer simple passwords that are easy to remember (like "`easy`"), but such passwords are often insecure. Some sites use random computer-generated passwords (like "`mdsduiep`"), but users have a hard time remembering them and sometimes leave them written on notes stuck to their computer.

One potential solution is to generate "pronounceable" passwords that are relatively secure but still easy to remember.

Alice is developing such a password generator. You work in the quality control department, and it's your job to test the generator and make sure that the passwords are acceptable. To be acceptable, a password must satisfy these three rules:

1. It must contain at least one vowel.
2. It cannot contain three consecutive vowels or three consecutive consonants.
3. It cannot contain two consecutive occurrences of the same letter, except for "`ee`" or "`oo`".

For the purposes of this problem, the vowels are '`a`', '`e`', '`i`', '`o`', and '`u`'; all other letters are consonants.

Note that these rules are not perfect; there are many common/pronounceable words that are not acceptable.

Input

The input consists of no more than 50 potential passwords, one per line, followed by a line containing only the word "`end`" that signals the end of the file and shouldn't be processed. Each password is at least one and at most twenty letters long and consists only of lowercase English letters.

Output

For each password, print 1 if it is acceptable and 0 otherwise.

Example

standard input	standard output
my	0
ic	1
pc	0
myicpc	0
contest	1
problem	1
biir	0
beer	1
end	

Problem P. Bananas

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

It is well known story about five sailors and a monkey who were shipwrecked on an island. They spent the first day gathering bananas. During the night, one sailor woke up and decided to eat his share of the bananas. He divided them into five piles. One banana was left over so he gave it to the monkey, then eats his share and went back to sleep.

Soon a second sailor woke up and did the same thing. After dividing the bananas into five piles, one banana was left over which he gave to the monkey. He then eats his share and went back to bed. The third, fourth, and fifth man followed exactly the same procedure.

The next morning, after they all woke up, they divided the remaining bananas into five equal shares. This time no bananas were left over.

An obvious question is "how many bananas did they originally gather?" There are an infinite number of answers, but the lowest of these is 3121. Suppose we turn the problem around. If we know the number of bananas that were gathered, what is the maximum number of sailors (and one monkey) that could have been shipwrecked if the same procedure could occur?

Input

The input consists of no more than 20 test cases. Each test case consists of one integer k ($2 \le k \le 10^9$) — the number of bananas gathered by a group of persons (and a monkey) that were shipwrecked. The input will be terminated by zero.

Output

For each test case, print the largest number of persons who could have participated in the procedure described above or -1 if this procedure is impossible.

Example

standard input	standard output
25 30 3121 0	3 -1 5

Problem Q. Cities

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

The map of the country can be represented as grid with r rows and c columns. Each cell of the grid is either empty or there is a building in it. We call two buildings adjacent if they share an edge or vertex. We say there is a path between two buildings if there is a sequence of the buildings connecting them such that each element of this sequence is adjacent to its previous and next elements.

We will define a *city* as maximal set of buildings such that each pair of buildings in the set has a path to each others. Given the map, find the number of cities.

Input

First line of the input contains one integer $1 \le T \le 20$ — number of the test cases.

First line of each test case contains two integers r and c — the number of rows and columns, respectively ($1 \le r, c \le 100$). Each of next r lines consists of exactly c characters.

Character '`X`' means that there is a building in that cell and character '`.`' shows that its an empty cell.

Output

For each test case output the number of cities.

Example

standard input	standard output
2 2 3 .X. X.. 4 4 ...X X... ...X .XXX	1 3

Problem R. Recover The Permutation

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

Teacher gave a number of permutations of the n integers 1, 2, ..., n to her students. For each integer i, $(1 \le i \le n)$, she asks the students to write down the number of integers greater than i that appear before i in the given permutation. This number is denoted a_i . For example, if $n = 8$ and the permutation is 2, 7, 3, 5, 4, 1, 8, 6, then $a_1 = 5$ because there are 5 numbers $(2, 7, 3, 5, 4)$ greater than 1 appearing before it. Similarly, $a_4 = 2$ because there are 2 numbers (7, 5) greater than 4 appearing before it.

Rikka, one of the students in the class, is studying for the final exams now. She found out that she has lost the assignment questions. She only has the answers (the a_i's) but not the original permutation. Can you help her determine the original permutation, so she can review how to obtain the answers?

Input

The input starts with a line containing the integer n ($n \le 500$). The next line give the values of a_1, a_2, ..., a_n.

Output

Print a line specifying the original permutation. Adjacent elements of a permutation should be separated by a space. Note that some cases may require you to print lines containing more than 80 characters.

Examples

standard input	standard output
10 1 8 6 6 2 4 0 0 0 0	7 1 8 5 9 10 6 3 4 2