Problem A. Box Game

Input file: `standard input`
Output file: `standard output`

This problem is interactive — here you must implement an algorithm to play a game. Note that your program has to communicate via standard input and standard output (i.e., from the keyboard and to the screen). After writing each line, be sure to use the stream flushing function. Otherwise you risk leaving part of your output in I/O buffer. For example, you must use the `fflush(stdout)` function in C++, call `System.out.flush()` in Java, and `flush(output)` in Pascal.

Alice and Bob play the following box game. There are n boxes on the table in a row, numbered from 1 to n from left to right. For each box Alice and Bob know its capacity c_i — the maximal number of balls which can be placed in it. So the i-th box can contain any number of balls between 0 and c_i, inclusive. Initially the i-th box contains s_i balls.

Alice starts and the players take turns alternately. In a turn, a player should add a single ball into any box or take a single ball from any box. Obviously, a player can't place a ball into a box (say, i) if it already contains c_i balls. Similarly, a player can't take a ball from a box if the box is empty. There is one more rule: a player can make only such a move that leads to the situation that didn't happen before. In other words, a player can't make a move that leads to the game situation which already happened before. Obviously, it makes the game finite. A player loses if (s)he can't make a move.

You may assume that the balls are indistinguishable and the players have as many balls as may be needed in a game.

Write a program to play this game against the jury program. Your program should always win. You choose if you play for Alice or for Bob in each test case.

Input

The input contains multiple test cases. The first line of each test case contains n ($1 \le n \le 15$) — the number of the boxes. The second line contains the sequence of integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10\,000$) — the boxes' capacities. It is guaranteed that the number of all possible states in the game doesn't exceed 50 000, that is, $\prod_{i=1}^{n}(c_i + 1) \le 50\,000$. The third line contains the integers $s_1, s_2, \ldots, s_n$ ($0 \le s_i \le c_i$) that give the initial arrangement of the balls.

A move description is a line containing an integer number p ($1 \le p \le n$) and a character '`+`' or '`-`', separated by a space. Character '`+`' means that a player places a ball into the p-th box, while character '`-`' means that a player removes a ball from the p-th box.

The rest of the test case contains lines with move descriptions of the opponent, one description per line. If you play for Alice, write your first move, read the first opponent's move, then write your second move and so on. If you play for Bob, the first move is made by the opponent. So you should read the first opponent's move, then write your first move, read the second move of the opponent and so on. When your opponent can not make a move, he writes '`0 ?`' instead of his move and the test case ends.

The input ends with $n = 0$. You should not process this test case.

Output

For each test case, print "`Alice`" if you want to move first, and "`Bob`" otherwise. Then print move descriptions one per line. After writing each line, be sure to use the stream flushing function.

Examples

standard input	standard output
2 1 1 1 1 2 - 0 ? 1 4 2 1 + 0 ? 0	Alice 1 - 1 + Bob 1 +

Problem B. Circle of Stones

Input file: standard input
Output file: standard output

There are n stones on the table in a circle, each of them has a color. Colors are represented by lowercase English letters, so there are 26 possible colors. Different stones can have equal colors.

The circle of stones is called *motley* if every two neighboring stones have different colors. Stones in a circle are considered neighboring if there are no other stones between them in at least one direction. For example, the 1st and the 2nd stones are neighboring, as well as the 2nd and the 3rd, or the n-th and the 1st.

You are allowed to take away any (possibly empty) sequence of consecutive stones. And you can do this operation only once.

Determine for each k ($0 \le k < n$) if it is possible to take away k stones so that the resulting circle would be motley.

Input

The input contains several test cases. Each test case is a single line, which is a description of n stones lying on the table ($1 \le n \le 10^6$). This line contains only lowercase English letters.

Output

For each test case, display the case number and a string of n digits. The k-th digit (0-indexed) must be "1" if it is possible to take away a segment of k consecutive stones so that the resulting circle would be motley. Otherwise the k-th digit must be "0".

Examples

standard input	standard output
rrg rrrrr brbg abab	Case 1: 011 Case 2: 00001 Case 3: 1111 Case 4: 1011

Problem C. Expression with Sets

Input file: standard input
Output file: standard output

Expression with sets is an expression which has as operands some sets of integers between 0 and 999 999 999, inclusive. More formally, all these sets are subsets of the universal set $U = \{x : 0 \le x < 10^9\}$. In the expression, each set is denoted as a possibly empty list of different integers separated with commas (“,”). Every such list is enclosed in braces (“{” and “}”). For example, these are valid sets: “{}”, “{0}”, “{1,2,3,4}”, “{999999999,9,3}”.

There are three possible operations with sets:

- Complement. Denoted as “!”. For any set $A \subseteq U$, we have $!A = U \setminus A = \{x : (x \in U) \ \& \ (x \notin A)\}$. For example, !{1,5,2}={0,3,4,6,7,8,...,999999999}.
- Intersection. Denoted as “&”. For any two sets $A, B \subseteq U$, we have $A\&B = \{x : (x \in A) \ \& \ (x \in B)\}$. For example, {0,1,2,3,4}&{1,5,2}={1,2}.
- Union. Denoted as “|”. For any two sets $A, B \subseteq U$, we have $A|B = \{x : (x \in A) \vee (x \in B)\}$. For example, {0,1,2,3,4}|{1,5,2}={0,1,2,3,4,5}.

Here, the operations are listed in order from the highest priority to the lowest priority. The expression may also contain some brackets (“(” and “)”). Note that according to operations’ priorities, $A|!B\&C = A|((!B)\&C)$.

You are given an expression with sets. Evaluate it and print the number of elements in the resulting set.

Input

The input contains several test cases. Each test case is a single line which is a single expression with sets. The length of the expression is between 1 and 10^7 characters. The total number of operations (i.e. the number of characters “!”, “&”, “|”) does not exceed 10^6.

It is guaranteed that every expression is valid (i.e. it satisfies all the above descriptions) and the input doesn’t contain any spaces.

Output

For each test case, display the case number and the number of elements in the resulting set.

Examples

standard input	standard output
!{1,5,2} {0,1,2,3,4}&{1,5,2} {0,1,2,3,4}|{1,5,2} !{1,2,3}|({1}) ({1}|{2})&{1}	Case 1: 999999997 Case 2: 2 Case 3: 6 Case 4: 999999998 Case 5: 1

Problem D. Heating System

Input file: `standard input`
Output file: `standard output`

The house heating system consists of n nodes and m bidirectional pipes connecting pairs of nodes. The system is designed so that the hot water is circulating through the pipes and the amount of water coming to each node (in a unit of time) is equal to the amount of water leaving the node. The only exceptions are two special nodes: *source* (has outgoing flow) and *sink* (has incoming flow). All other nodes satisfy the restriction that the amount of flow into a node equals the amount of flow out of it.

Each pipe is described by two properties: *capacity* and *friction coefficient*. The capacity of the i-th pipe is denoted as c_i and stands for the maximal amount of water that can pass through it in a unit of time. Water can move in any of two directions in a pipe. The friction coefficient of the i-th pipe is denoted as p_i. The friction in the i-th pipe depends on p_i and the current flow f_i through it, the friction value is equal to $p_i \cdot f_i^2$.

It is known that water passes the heating system in such a way that the total flow is maximal. It means that the amount of the water passing through the system in a unit of time is maximal. Among all the ways satisfying the first property, the flow passes through the system in such a way that the sum of all frictions is minimal.

Write the program which finds such maximal flow that the total friction is minimal.

Input

The input contains several test cases. Each test case starts with a line containing two positive integers n and m ($2 \le n \le 50, 1 \le m \le 100$) as described above. The following m lines contain four integer numbers x_i, y_i, c_i and p_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$; $1 \le c_i, p_i \le 50$), where x_i and y_i are the numbers of the nodes connected by the i-th pipe, c_i and p_i are the capacity and friction coefficient of the i-th pipe. There is no more than one pipe between any pair of nodes. The nodes are numbered from 1 to n.

The first node is a source, the last node is a sink. It is also guaranteed that the sink is reachable from the source by the pipes.

Output

For each test case, display the case number, the maximal flow value and the minimal total friction. Display the flow and the friction with at least 3 digits after the decimal point. Print the flows $f_1, f_2, \ldots, f_m$ through the pipes 1, 2, ..., m. Positive f_i stands for the flow direction from the x_i to y_i, while a negative value stands for the opposite direction. Display each f_i in the output with at least 9 digits after the decimal point.

Examples

standard input
5 5 2 1 1 1 2 3 1 1 1 4 1 1 4 3 1 1 3 5 1 1 3 1 1 3 13 17
standard output
Case 1: 1.0000000000 2.0000000000 -0.5000000000 0.5000000000 0.5000000000 0.5000000000 1.0000000000 Case 2: 13.0000000000 2873.0000000000 13.0000000000

Problem E. Islands

Input file: standard input
Output file: standard output

Sergey opened his eyes and found himself in a game. He was on a small island with other boys and girls fighting with wooden swords and trying to capture the neighbouring islands. There were exactly forty islands in the boundless ocean, and each of them was connected with three neighboring islands. The game has only three main rules: not to play after the bridges are raised, not to play in the giveaway and not to look up at the sunset.

When Sergey is so close to panic, he tries to think about mathematical problems. That's why it is not important for him that he can be killed at any moment, who is the author of this book, what is the goal of this silly game and how he can escape. He is focused on the only question: is all that possible from the geometrical point of view? Since Sergey likes generalizations, he came to the following problem.

Consider the archipelago containing $n \times m$ islands which are located in all the points with integer coordinates (x, y), $0 \le x < n$, $0 \le y < m$. The task is to connect some pairs of the islands with bridges (which are straight segments) in such a way that the following conditions are satisfied:

- every island is connected by bridges with exactly p other islands;
- there is at most one bridge between each pair of islands;
- no bridge passes through islands different from its ends;
- no two bridges intersect in points different from their common end;
- the total length of the bridges is minimal possible.

Input

The input contains several test cases. Each test case consists of three integers n, m and p ($1 \le n, m \le 100$, $1 \le p \le 8$) which represent the archipelago sizes and the number of neighbors each island is connected to by bridges.

Output

For each test case, display the case number. Then, if the problem does not have a solution, print "-1". Otherwise, print a real number — the minimum total length of bridges. An absolute or relative error should not exceed 10^{-6}. The second line should contain the number of the bridges. The following lines should contain the descriptions of the bridges, one per line. Every bridge should be described by four integers "x_1 y_1 x_2 y_2" which represent the coordinates of the islands connected by the bridge. The order of the bridges and the order of points in each pair are not important. If there are multiple solutions, print any of them.

Examples

standard input	standard output
2 2 2 2 2 3	Case 1: 4.0000000000 4 1 1 1 0 1 1 0 1 1 0 0 0 0 1 0 0 Case 2: -1

Problem F. Longest Two Graphs Common String

Input file:	`standard input`
Output file:	`standard output`

You are given two directed graphs G_1 and G_2. The vertices of both graphs are labeled with lowercase English letters. Different vertices can have the same labels, each vertex is labeled with exactly one letter.

Moving along a path in a graph one may pronounce a string. So each path $v_1, v_2, \ldots, v_k$ in a graph (where (v_i, v_{i+1}) is an arc for each $1 \le i < k$) generates a string "$l_{v1} l_{v2} \ldots l_{vk}$", where l_v is the label of the vertex v. A path may contain cycles, it can degenerate to a single vertex.

One may pronounce set of strings using G_1 and set of strings using G_2. What is the length of the longest common string in two sets? Write a program which will find the required length and the corresponding paths in both graphs.

Input

The input contains several test cases. Each test case consists of description of G_1 and description of G_2. Each description starts with the line containing integers n and m ($1 \le n \le 500$, $0 \le m \le 4000$), where n is the number of vertices and m is the number of arcs. The next line contains a string of exactly n lowercase English letters, the i-th letter in the string stands for the label of the i-th vertex. The following m lines contain arcs, one arc description per line. Each description is a pair of integers x_j, y_j ($1 \le x_j, y_j \le n$) — the starting and finishing endpoints of the j-th arc. Graphs can contain loops and multiple arcs.

The test cases are separated with a blank line.

Output

For each test case display the case number and the length of the longest common string. If the longest common string is infinite, display "`-1`" as the length. In the case of finite longest common string, display two additional lines. Lines should contain vertex indices along the corresponding path in G_1 and G_2. If there are multiple answers, display any of them.

Examples

standard input	standard output
6 5 abacab 1 2 2 3 3 4 4 5 5 6 5 5 aabac 1 2 2 3 3 4 4 5 5 1 1 1 a 1 1 2 2 aa 1 2 2 1	Case 1: 5 1 2 3 4 5 2 3 4 5 1 Case 2: -1

Problem G. Lyndon Words

Input file: standard input
Output file: standard output

Lyndon word is a string that is strictly smaller in lexicographic order than all of its cyclic shifts. For example, "abbac" is a Lyndon word, but "abbab" is not.

You are given two positive integers n and m. Consider all Lyndon words not longer than n characters and containing letters from the first m letters of English alphabet. Let's numerate them in lexicographic order starting from 1. Write a program which will display the part of this list from the number l to the number r inclusively.

Input

The input contains several test cases. Each test case contains four positive integers n, m, l and r ($1 \le n \le 30$; $2 \le m \le 26$; $1 \le l \le r \le 10^7$). It is guaranteed that $r - l < 10^5$ and there are at least r Lyndon words not longer than n containing letters from the first m letters of English alphabet.

Output

Display the test case number and the required list of words, one word per line. Order words lexicographically.

Examples

standard input	standard output
4 3 7 12 4 3 31 32	Case 1: aac aacb aacc ab abac abb Case 2: bccc c

Note

There are 32 Lyndon words for $n = 4$ and $m = 3$: a, aaab, aaac, aab, aabb, aabc, aac, aacb, aacc, ab, abac, abb, abbb, abbc, abc, abcb, abcc, ac, acb, acbb, acbc, acc, accb, accc, b, bbbc, bbc, bbcc, bc, bcc, bccc, c.

Problem H. Temperature

Input file: standard input
Output file: standard output

It is autumn in Berland and that means another flu epidemic is coming.

The teacher gave each of n students a thermometer and asked to measure their temperature. But the problem is, some kids are too lazy to do it and they just lie about their temperature.

Each student knows which of their classmates tells the truth and which one lies. The ones who tell the truth will simply say integer t_i — the value shown by their thermometer. The others act as follows. Each of them:

- chooses some *non-empty* subset of students R_i, who told the teacher the truth;
- picks a rounded down to the closest integer arithmetic mean of the values they stated: $a = \left\lfloor \frac{\sum_{j \in R_i} t_j}{|R_i|} \right\rfloor$;
- adds an arbitrary integer from $-x_i$ to x_i, inclusive: $t_i = a + random(-x_i, x_i)$;
- tells the resulting value t_i to the teacher.

Numbers x_i are individual for each student and are known to everybody.

After all students said their t_i values, the teacher wondered: what minimum number of kids in the class could have told the truth?

Input

The input contains several test cases. Each test case starts with a line containing a single integer n ($1 \le n \le 20$) — the number of kids in the class. Next n lines contain pairs of integers t_i and x_i ($1 \le t_i \le 10^6$; $0 \le x_i \le 10^6$) — the temperature the i-th student told and the maximum deviation from the mean if the student didn't tell the truth. The numbers on the lines are separated by a space.

Output

For each test case, display the case number and a single positive integer — the minimum number of kids who could have told the truth.

Examples

standard input	standard output
3 1 0 2 2 3 1 4 10 5 20 5 15 1 25 6	Case 1: 2 Case 2: 2

Note

In the first case, the first and third students told the truth and the second student told the teacher their mean temperature $\frac{1+3}{2} = 2$.

In the second case only the first and second students told the truth. The third student took the arithmetic mean of the first two and the arbitrary number he added was equal to 0, that is, $\frac{10+20}{2} + 0 = 15$. The fourth student took the temperature of the second student and the arbitrary number got equal to 5, that is $\frac{20}{1} + 5 = 25$.

Problem I. Editor 2

Input file: `standard input`
Output file: `standard output`

Vasya is pressing the keys on the keyboard reluctantly, squeezing out his ideas on the classical epos depicted in Homer's Odysseus... How can he explain to his literature teacher that he isn't going to become a writer? In fact, he is going to become a programmer. So, he would take great pleasure in writing a program, but none — in writing a composition.

As Vasya was fishing for a sentence in the dark pond of his imagination, he suddenly wondered: what is the least number of times he should push a key to shift the cursor from one position to another one?

Let's describe his question more formally: to type a text, Vasya is using the text editor. He has already written n lines, the i-th line contains a_i characters (including spaces). If some line contains k characters, then this line overall contains $(k+1)$ positions where the cursor can stand: before some character or after all characters (at the end of the line). Thus, the cursor's position is determined by a pair of integers (r, c), where r is the number of the line and c is the cursor's position in the line (the positions are indexed starting from one from the beginning of the line).

Vasya doesn't use the mouse to move the cursor. He uses keys "Up", "Down", "Right" and "Left". When he pushes each of these keys, the cursor shifts in the needed direction. Let's assume that before the corresponding key is pressed, the cursor was located in the position (r, c), then Vasya pushed key:

- "Up": if the cursor was located in the first line ($r = 1$), then it does not move. Otherwise, it moves to the previous line (with number $r - 1$), to the same position. However, if the previous line was too short, that is, the cursor couldn't occupy position c there, the cursor moves to the last position of the line with number $r - 1$;
- "Down": if the cursor was located in the last line ($r = n$), then it does not move. Otherwise, it moves to the next line (with number $r + 1$), to the same position. However, if the next line was too short, that is, the cursor couldn't occupy position c there, the cursor moves to the last position of the line with number $r + 1$;
- "Right": if the cursor can move to the right in this line ($c < a_r + 1$), then it moves to the right (to position $c + 1$). Otherwise, it is located at the end of the line. If the current line is the last one ($r = n$), the cursor doesn't move anywhere when Vasya presses the "Right" key. Otherwise, it moves to the leftmost position in the next line (with number $r + 1$).
- "Left": if the cursor can move to the left in this line ($c > 1$), then it moves to the left (to position $c - 1$). Otherwise, it is located at the beginning of the line. If the current line is the first one ($r = 1$), the cursor doesn't move anywhere when Vasya presses the "Left" key. Otherwise, it moves to the rightmost position in the previous line (with number $r - 1$).

You've got the number of lines in the text file and the number of characters written in each line of this file. Also you are given m queries. For each query, find the least number of times Vasya should push the keys described above to move the cursor from position (r_1, c_1) to position (r_2, c_2).

Input

The input contains several test cases. Each test case starts with a line containing two integers n and m ($1 \le n \le 10^6$, $1 \le m \le 10$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$), separated by single spaces. Each of the following m lines contains four integers r_1, c_1, r_2, c_2 ($1 \le r_1, r_2 \le n$, $1 \le c_1 \le a_{r_1} + 1$, $1 \le c_2 \le a_{r_2} + 1$).

Output

For each test case, display the case number followed by the answers to the queries. For each query, display

the minimum number of times Vasya should press a key to move the cursor from position (r_1, c_1) to position (r_2, c_2).

Examples

standard input	standard output
4 1 2 1 6 4 3 5 4 2 5 2 3 5 4 3 6 2 1 5 6 1 1 5 7	Case 1: 3 Case 2: 7 9

Problem J. Tourist Problem

Input file: standard input
Output file: standard output

"One does not simply set up a tourist tent", Gennady said.

As you probably know Gennady is a tourist. Now he is in a forest consisting of n trees. No three trees are on the same line. It is evening and Gennady wants to set up a tourist tent.

To set up a tourist tent he should choose k trees forming a convex polygon. There is an additional restriction: there must be exactly one tree inside the polygon to be used as a pole to maintain the tent.

Gennady can't decide which trees will be used. He decides to analyze each valid set of k trees. It is not a problem for him to count them.

Imagine that you are a tourist and count the number of valid tree sets to set up a tent.

Input

The input contains several test cases. Each test case starts with a line containing two positive integers n, k ($1 \le n \le 250$, $3 \le k \le 10$) as described above. The following n lines contain a pair of integer coordinates (x_i, y_i) of the i-th tree. The absolute values of the coordinates never exceed 10^4. No three trees are located on the same line. The given n points are distinct.

Output

For each test case, display the case number and the required number of valid polygons. It is guaranteed that the answer fits in signed 64-bit integer type.

Examples

standard input	standard output
5 3 0 10 10 0 10 10 0 0 1 2 5 4 0 10 10 0 10 10 0 0 1 2	Case 1: 2 Case 2: 1

Problem K. Ant versus Woodpecker

Input file:	`standard input`
Output file:	`standard output`

The Ant lives near a tree, or, to be precise, near the tree root. The tree consists of n nodes and $n-1$ branches, each branch connects a pair of nodes. The Ant can reach any node walking only along tree branches. The tree nodes are numbered from 1 to n, and the root is the node number one.

From time to time the Ant does a sally for obtaining a food. Before doing each sally, the Ant finds out $v_1, v_2, \ldots, v_k$ — numbers of nodes containing the food. The Ant wants to visit all these nodes. He starts his way in the tree root, then, moving along the tree branches, visits all nodes with food in arbitrary order, and finally returns back to the tree root.

The Ant's life is not easy, as he is always in danger: the Woodpecker hunts him! Namely, if the Ant visits some node *more than two times* on his way, then he risks being noticed by the Woodpecker. If the Ant is noticed by the Woodpecker, then he starts running away. In that case the Ant interrupts his sally and returns to the root using the shortest path. The longer is the distance from the root to the node where he was noticed, the worse is the situation. That's why the Ant wants to find such path visiting all nodes with food which will minimize this escape distance in the worst case.

Write a program that will determine the maximum distance from the root to the node where the Ant can be noticed by the Woodpecker if the Ant chooses the optimal path. You will be given descriptions of several sallies, and you will have to find the optimum value for each of them.

Additionally, the tree is not static:

- new nodes may be added to the tree (i.e. a node and a branch connecting it to one of the existing nodes are created),
- nodes can be deleted (i.e. some branch breaks off, and a whole subtree disappears).

Input

The input consists of one or more test cases.

Each test case starts with a line containing integer n ($2 \le n \le 10^5$) — the number of nodes in the tree in the beginning. The second line contains the initial tree description: $n-1$ integers $p_2, \ldots, p_n$, where p_i ($i = 2 \ldots n$) is the number of node from which the Ant will go to the i-th node if he moves away from the root. It is guaranteed that the input describes a correct tree with nodes numbered from 1 to n.

Then an integer m follows — the number of queries ($1 \le m \le 10^5$). Each query describes one of the three events: the Ant's sally, creation of a node, or deletion of a subtree. Each of the following m lines describes a query using one of the following formats:

- Query of the first kind: "**? k v_1 v_2 ... v_k**".

  The Ant has to visit k nodes with food with given distinct numbers $v_1, v_2, \ldots, v_k$ (the Ant may vary the order of visiting these nodes). As an answer for this query you have to output the longest escape distance for the Ant if he chooses the optimal path (or output -1 if the Ant may finish his path without a risk of being captured by the Woodpecker).

  It is guaranteed that the nodes numbered $v_1, v_2, \ldots, v_k$ existed just before executing this query.

- Query of the second kind: «**+ x**».

  A new node, connected with one of the existing nodes, appears in the tree. The new node gets numbered as the smallest positive integer not occupied currently by any other node. This new node gets connected with a branch to the existing node number $x + y$, where number y is the result of

the last query of the first kind. It is guaranteed that at least one query of the first kind has already occurred, and that the node number $x + y$ existed just before executing this query. The number x in the input may be an **arbitrary** integer satisfying restrictions above.

- Query of the third kind: «$-$ v».

  The node number v with its subtree has to be deleted from the tree. A subtree is a set of nodes which can't be reached from the root without visiting node v. Pay attention to that the numbers of deleted nodes are being freed, i.e. these numbers become available for the new nodes in subsequent queries of the second kind. It is guaranteed that the node number v existed just before executing this query. Also it is guaranteed that $v > 1$, i.e. the tree root cannot be deleted.

Sum of k over all queries of the first kind in one test case does not exceed 10^5.

Output

For each test case, output a line containing the test number and the answers for all queries of the first kind. It is guaranteed that each test case contains at least one query of the first kind. The queries should be processed in the order they are given in the input.

Examples

standard input	standard output
3 1 1 4 ? 2 2 3 + 2 + 2 ? 3 2 4 5 2 1 4 ? 2 1 2 ? 1 2 - 2 ? 1 1 5 3 1 1 3 4 ? 2 2 5 - 2 + 0 ? 2 2 5	Case 1: 0 1 Case 2: -1 -1 -1 Case 3: 1 0