icpc | europe contests

The 2023 ICPC Central Europe Regional Contest

Official Problem Set

A – Attendance

Time limit: 8 s *Memory limit: 128 MiB*

An ambitious university student has enrolled in just about every possible course. Unfortunately, the courses require mandatory attendance. He has decided to visit the university campus where the lectures are held several times a day. He will join every lecture that is running at that moment, sign the attendance sheet, and immediately leave the campus due to other obligations. He will return later that day, when he will repeat this process to sign attendance sheets at other lectures and so on until his name is on attendance sheets of all lectures.

As if this was not problematic enough, the student faces another obstacle: the schedule of the lectures keeps changing. Some lectures are added and some are canceled. The student has to keep adjusting his visiting schedule of the university to sign attendance sheets at all lectures.

Write a program that will start with an empty schedule of lectures and read sequential modifications, which are either an addition or removal of a single lecture. For every modification, output the minimum number of visits that the student has to make to sign attendance sheets at all lectures that are currently on the schedule.

Input data

The first line contains the number of modifications N, which are given in the following N lines. An addition of a lecture is described with two space-separated integers A_i and B_i, which represent a lecture that is running from A_i to B_i (including both bounds). The lectures are numbered as they are added, sequentially from 1 onwards. A negative number X_i represents a removal of lecture with the number $-X_i$.

Input limits

- $1 \leq N \leq 300\,000$
- $0 \leq A_i \leq B_i \leq 10^9$
- Every number of the lecture for removal X_i will be valid – it will exist in the schedule at that moment.
- Note the memory limit.

Output data

For every modification output a single line with the minimum number of required visits for the current schedule of lectures.

Example

Input	Output
12 2 2 17 26 -2 12 21 0 0 19 21 16 22 14 20 15 19 13 14 -4 13 17	1 2 1 2 3 3 3 3 3 4 3 3

Comment

The first lecture to be added is $[2, 2]$ and is given number 1. Next added lecture is $[17, 26]$ with number 2. It is removed immediately afterwards, which is indicated by -2 in the input. The following added lecture is $[12, 21]$, which is given number 3 and so on.

B – Ball Passing

Time limit: 1 s *Memory limit: 256 MiB*

A group of students has just finished their math lesson and they're heading out for physical education. Their teacher has asked them to arrange themselves in a circle. After several minutes of busy moving around the court they have finally managed to position themselves so that they form a *strictly convex polygon*. They might not lie on the circle, but the teacher is happy to at least get some structure.

There is an even number of boys and an even number of girls in this group of N students. They will practice ball passing in pairs, therefore the teacher has to pair them up. The teacher will pair boys among themselves and the same for girls.

The school administration has decided to address the decline in physical performance of their students. Therefore, they have implemented a quality measure for ball passing practice, which is the total distance traveled by the balls in a single round of ball passes between each pair. Help the teacher pair up the students in a way that will maximize this measure.

Input data

The first line contains the number of students N. The second line contains a string S of length N, which describes the students along the perimeter of the polygon with a character "`B`" for a boy and "`G`" for a girl. The following N lines provide the locations of students with space-separated integer coordinates X_i and Y_i in the same order as they are described in the string S.

Input limits

- $2 \leq N \leq 50$
- The number of boys and girls will both be even. Note that one of them can be zero.
- The coordinates X_i and Y_i won't exceed 10 000 by absolute value.

Output data

Output the maximum ball passing distance that can be obtained by pairing up the students appropriately. The solution will be considered correct if the relative or absolute error compared to the official solution is within 10^{-6}.

Examples

Input	Output
4 BGBG 0 0 0 1 1 1 1 0	2.828427125

Input	Output
4 GGBB 0 0 0 1 1 1 1 0	2

Input	Output
12 GBGBBGBBBBGB 0 -15 6 -14 19 -5 17 7 11 12 1 15 -9 13 -15 10 -17 8 -19 4 -16 -9 -13 -11	186.529031603

C – Cakes

Time limit: 2 s *Memory limit: 256 MiB*

Your local cake shop is making a business plan for the next few months. The bakers have C different recipes, each requiring their own set of ingredients and tools. During the baking, the ingredients are consumed, but the tools are not and can be reused for other recipes. Currently, the bakery has no ingredients or tools – they were all destroyed in the recent floods or taken away by the tax bureau.

The son of the main chef managed to convince everyone to only bake each type of cake once. Individuals on the internet are supposedly happy to pay extra to be the only owners of their own unique Nutty-Fudge Tart (NFT). In fact, the son has already gone ahead and estimated how much money they can earn for each type of cake. Now bakers are looking at each other, trying to figure out which types of cake to prepare for maximum profit. You are given the costs of all ingredients, tools, and prices of cakes. Your task is to determine how much profit the bakers can make.

Input data

The first line contains three integers: G, C, and T, the number of ingredients, the number of recipes, and the number of different tools in them, respectively. The second line contains C space-separated integers $c_1, \ldots, c_C$, the prices of each cake. The third line contains G space-separated integers $g_1, \ldots, g_G$, representing the prices of each ingredient. The fourth line contains T space-separated integers $t_1, \ldots, t_T$, representing the prices of all tools.

This is followed by C lines, each containing G space-separated integers $a_{i,j}$, corresponding to the amount of ingredient j in cake i.

Finally, this is followed by C lines of the following format: the i-th row starts with an integer n_i, the number of tools required for i-th cake. This is followed by n_i space-separated integers $b_{i,k}$, indicating that we need tool $b_{i,k}$ to prepare cake i (listed tools are distinct).

Input limits

- $1 \le G, C, T \le 200$
- $0 \le c_i, t_i \le 10^9$
- $0 \le g_j, a_{i,j} \le 10^8$
- $0 \le n_i \le T$
- $1 \le b_{i,k} \le T$

Output data

Print a single number: the maximum profit that the cake shop can make.

Example

Input

```
5 3 4
14 18 21
1 2 3 1 2
5 6 3 10
0 0 1 2 0
1 2 0 1 2
5 2 1 0 0
2 1 2
2 2 3
2 3 4
```

Output

```
3
```

Comment

The maximum profit is made by baking cakes 1 and 2, but not cake 3.

D – Drying Laundry

Time limit: 3 s *Memory limit: 256 MiB*

Harry the Beaver runs a hotel and has to wash bed sheets every Sunday night for the next Q weeks until the tourist season ends. On week j, he has N freshly washed bed sheets that he wants to dry by hanging them on two parallel clotheslines of length L_j each. The sheets can be hung next to each other but must not overlap. Each sheet is d_i units wide and rather long, therefore he will always orient it so that it will take up d_i units of the line when hung to dry. The sheets have different drying times that are not related to their sizes because of different materials. Thus, the i-th sheet needs t_i^{slow} minutes to dry. However, if it is hung over both lines at the same time, it dries quicker in t_i^{fast} minutes, but also takes up space on the other line. To avoid smelly sheets, Harry the Beaver has to start drying all of them immediately after washing, i.e. all sheets have to be hung simultaneously.

Harry the Beaver wants to go to sleep as soon as possible on Sundays, therefore, he asks you to help him determine the minimal required drying time for each week j, or inform him that it is impossible to finish drying the sheets that week.

Input data

The first line contains an integer N, the number of sheets, and an integer Q, the number of weeks until the end of the tourist season. The next N lines contain space-separated integers d_i, t_i^{fast}, and t_i^{slow}, which correspond to the width, the shorter drying time, and the longer drying time of the i-th sheet, respectively. The final Q lines the the input contain integers L_j, j-th of which represents the length of the clothesline for week j.

Input limits

- $1 \le N \le 3 \cdot 10^4$
- $1 \le Q \le 3 \cdot 10^5$
- $1 \le d_i \le 3 \cdot 10^5$ for all $1 \le i \le N$
- $1 \le t_i^{\text{fast}} \le t_i^{\text{slow}} \le 10^9$ for all $1 \le i \le N$
- $1 \le L_j \le 3 \cdot 10^5$ for all $1 \le j \le Q$

Output data

Print Q lines, with j-th of them containing the minimal required drying time for week j, or "`-1`" (without the quotes) if it is impossible to finish drying the sheets that week.

Example

Input	Output
3 3 1 2 2 1 1 4 2 3 100 3 1 4	4 -1 3

E – Equal Schedules

Time limit: 1 s *Memory limit: 256 MiB*

You are one of the people on-call for a high-availability service that offers users to solve programming tasks. As an organized team, you have an on-call schedule specifying who is responsible for the service at which time. A colleague sends you a new schedule, and you want to make sure that everyone has the same amount of on-call time as before, or print any differences.

The on-call schedule is specified with lines of form s_i e_i t_i, where s_i and e_i represent the start and end offsets of the on-call shift for a teammate t_i from some start hour.

Given a sample schedule

```
0 7 jan
7 14 tomaz
14 20 jure
20 24 jan
24 25 tomaz
25 26 jure
```

we can see that `jan` is on-call for the first 7 hours (hour 0, 1, 2, 3, 4, 5, and 6), `tomaz` for next 7, ... In total, `jan` is on-call for 11 hours, `tomaz` for 8 and `jure` for 7.

Input data

The input contains two schedules separated by a horizontal line `------`. Each schedule contains one or more lines of form s_i e_i t_i, where integers s_i and e_i specify that teammate t_i is on-call for hours from s_i up to and excluding e_i. A final line `======` is printed after the second schedule.

Input limits

For each schedule, the following holds:

- $s_1 = 0$
- $s_i < e_i$
- $s_{i+1} = e_i$
- $e_i \leq 1000$
- Name t_i will consist of lowercase letters from the English alphabet.
- $3 \leq |t_i| \leq 20$

Output data

Output the differences between two schedules, in form t_i $\pm d_i$, where d_i is the difference between the second and the first schedule for the teammate t_i. The output should be sorted alphabetically by teammates' names and teammates with no differences should be omitted, otherwise the difference should be printed with a `+` or a `-` sign. If no differences are found, print "`No differences found.`" (without the quotes).

Examples

Input

```
0 7 jan
7 14 tomaz
14 20 jure
20 24 jan
24 25 tomaz
25 26 jure
------
0 9 tomaz
9 20 jan
20 26 jure
======
```

Output

```
jure -1
tomaz +1
```

Input

```
0 7 nino
7 14 bgs
14 21 ines
------
0 7 ines
7 14 nino
14 21 bgs
======
```

Output

```
No differences found.
```

Input

```
0 3 vid
3 6 maks
6 9 janez
------
0 1 vid
1 2 vid
2 3 vid
3 4 maks
4 5 maks
5 6 maks
6 7 janez
7 8 janez
======
```

Output

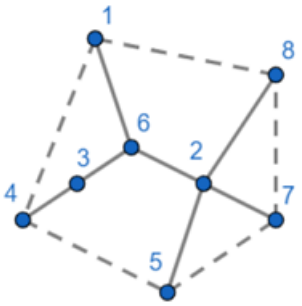
```
janez -1
```

F – Phylogenetics

Time limit: 5 s *Memory limit: 256 MiB*

A young biologist is studying evolutionary history and has come across phylogenetic trees. A phylogenetic tree shows evolutionary relationships among various biological species. It is presented in a planar embedding with its leaves arranged in a circular manner for a better visual presentation. We are dealing with an unrooted tree, where the leaves are nodes of degree 1. All nodes of the tree are colored, which makes distinguishing different species easier.

Our biologist is using graph visualization software which needs some help to produce a desired layout. Therefore, she has decided to add edges between adjacent leaves in the planar embedding. The tree has at least 3 leaves, which she connects in a cycle. The illustration below shows an example of such (uncolored) tree with additional edges between adjacent leaves indicated by dashed lines.



Now that the visualization is done, she is interested in the number of ways to color the nodes of this graph with K colors. Every pair of adjacent nodes should have a different color for easier visual recognition. Write a program that will read the description of her graph structure and compute the number of colorings.

Input data

The first line contains integers N, M and K. The edges of the graph are given in the following M lines as pairs of endpoints A_i and B_i (the nodes of the graph are numbered from 1 to N). All integers in the same line are separated by a space.

It is guaranteed that the graph was obtained from a planar embedding of a tree (acyclic connected undirected graph) by also connecting its leaves in a circular manner. The graph will not contain loops or parallel edges (i.e. multiple edges between the same pair of nodes).

Input limits

- $4 \leq N \leq 10^5$
- $1 \leq K \leq 10^5$

Output data

Output the number of colorings modulo $1\,000\,000\,007$.

Example

Input

```
8 12 3
2 5
3 6
2 6
5 4
4 1
1 6
7 5
2 7
3 4
2 8
7 8
1 8
```

Output

```
24
```

Comment

The example corresponds to the graph illustrated in the task.

G – Going to the Moon

Time limit: 1 s *Memory limit: 256 MiB*

Alice and Bob are playing a game in the sand outside their mansion. A circle representing the Moon is drawn somewhere, and they each also pick a place to stand (inside, on the edge, or outside the Moon). The goal of the game is that one of the players runs to the other as fast as possible, while also touching the Moon during the run.

Given the positions of the Moon, Alice, and Bob, find the length of the shortest path that starts at one of the players, touches (or crosses) the edge or the interior of the Moon, and ends at the position of the other player.

Input data

The first line contains an integer T, the number of test cases. It's followed by T lines, each containing 7 space-separated integers x_A, y_A, x_B, y_B, x_C, y_C, r, representing coordinates of Alice, $A = (x_A, y_A)$, Bob, $B = (x_B, y_B)$, the center of the circle, $C = (x_C, y_C)$, and its radius r.

Input limits

- $1 \leq T \leq 10^3$
- $-10^3 \leq x_A, y_A, x_B, y_B, x_C, y_C \leq 10^3$
- $0 \leq r \leq 10^3$

Output data

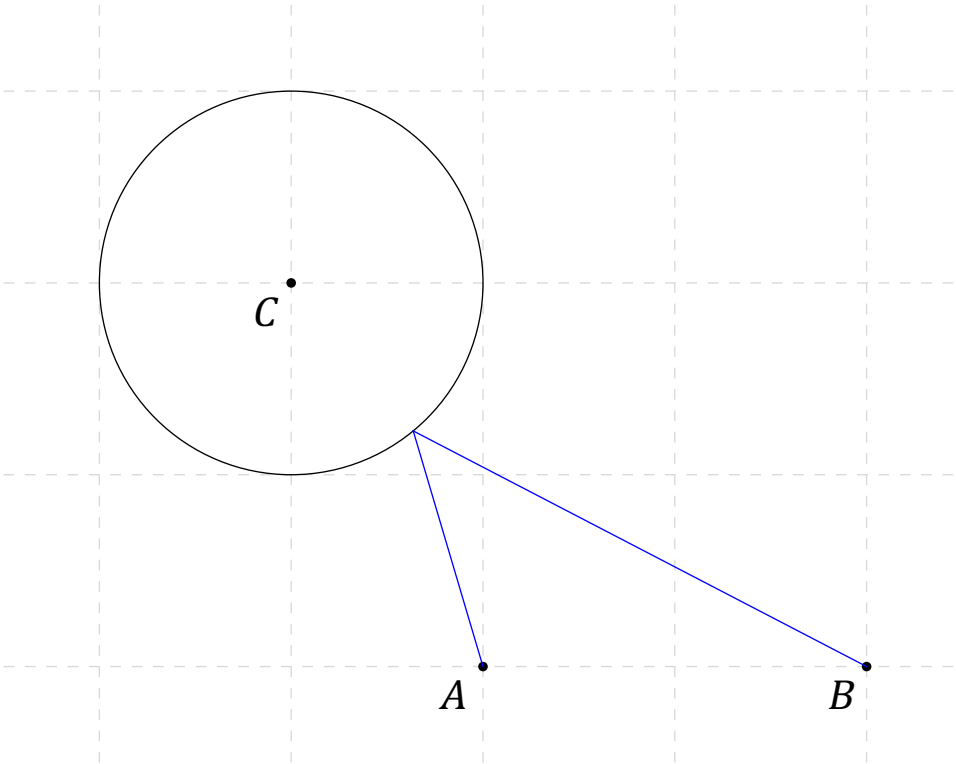
For each test case output a single decimal number representing the length of the shortest path from A to B that also touches at least one point inside or on the edge of a circle with the center C and radius r. The solution will be considered correct if the relative or absolute error compared to the official solution is within 10^{-6}.

Example

Input	Output
2 0 0 2 0 -1 2 1 5 0 3 0 2 0 2	3.9451754612261913 2

Comment

The solution for the first test case is shown in the picture.



H – Human Resources

Time limit: 4 s *Memory limit: 256 MiB*

You work at ECorp and your Human Resources department is migrating employee data from an on-premise system provided by Hooli to a new Cloud Native solution provided by a fresh startup Pied Piper. Sadly the new system does not yet have feature parity with the old one and they need your help to store and display the entire management structure. The system is lean and conscious of resource usage so they can only afford to increase their storage by two kilobits.

Input data

The first line of the input will be a command to execute either `ENCODE` or `DECODE`.

Encode

You will receive lines describing managers and their direct reports/subordinates. All lines will start with the name of the manager, followed by a colon, followed by a space-separated list of their direct reports ordered from their most to least favorite. Colons have a trailing space. No manager is listed before their manager, if they have one.

Decode

In the decode case you will be given the output that your program printed in the encode case: a list of all employee names in an arbitrary order, one per line, followed by a single line with a binary string B.

Input limits

- $2 \leq$ number of all employees at ECorp ≤ 600
- $|B| \leq 2048$
- Employee names are at most 10 characters long and consist of upper and lower case letters of the English alphabet.
- There is exactly one employee without a manager (the company CEO) and no employee has more than one manager.

Output data

Encode

Your program must first output the names of all the employees, one per line in an arbitrary order (this was a requirement from upper management). One additional line is dedicated for your encoding string B which must consist of ones and zeros and not exceed 2048 characters.

Decode

Output the original structure in the same format as it was originally given. The order of manager definitions does not have to be the same, but every one must come after their manager if they have one. However, the order of reports for any specific manager has to remain the same (from their most to least favorite).

Examples

Input

```
ENCODE
Janez: Josip Zofia
Zofia: Karolina
```

Output

```
Josip
Karolina
Janez
Zofia
00101100
```

Input

```
DECODE
Josip
Karolina
Janez
Zofia
00101100
```

Output

```
Janez: Josip Zofia
Zofia: Karolina
```

Comment

The encoding in the example above uses two consecutive characters for every person as they are ordered in the list. 11 denotes the CEO (Janez). 00 denotes a person at the second level of hierarchy. Their order in the CEO's list of direct reports is the same as in the encoded list of names (Josip and Zofia). Luckily, there are just two in this case. 10 denotes Zofia's report and 01 would denote Josip's reports, which he does not have.

I – Interactive Reconstruction

Time limit: 2 s *Memory limit: 256 MiB*

This is an interactive task where your program will communicate with a grader through standard input and output. Your task is to reconstruct a labelled tree with N nodes and $N-1$ edges. Nodes are labelled from 1 to N.

Your program is allowed to make a few queries of the following type: Your program should print a string of N characters, consisting only of zeros and ones, one corresponding to each node. The grader will return a sequence of N space-separated integers, the i-th representing the sum of the values (i.e. digits of the query string) of all neighbours of the i-th node. That is, if node j is a neighbour of node i, then the j-th digit of the query string counts towards the sum in the i-th number of the grader's answer.

See the example below for an illustration.

Input and output data

The first input line will contain the number N, the number of nodes in the tree.

Your program then has two options:

1. Print "`QUERY`" (without quotation marks), a space, and a string of N zeros and ones.
2. Print "`ANSWER`" (without quotation marks), a newline, and $N-1$ lines, each containing a pair of space-separated integers a, b, indicating that there exists an edge between nodes a and b.

If your program prints a query, this will be followed by the grader returning a line with N space-separated integers, one per node. If your program prints an answer, the grader will check the returned tree for correctness.

If there was a mistake in your queries, either due to incorrect formatting or due to an exceeded number of queries, the grader will print "`ERROR`" (no quotation marks) instead of the answer.

Important: Ensure that your program flushes its output after printing and correctly exits after printing the answer. It is up to you whether to implement the `ERROR` handling. Its purpose is to allow your program to exit gracefully and get a WA instead of a TLE verdict in the case of an error.

Input limits

- $2 \leq N \leq 3 \cdot 10^4$
- At most $2 \uparrow\uparrow 3 = 2^{(2^2)} = 16$ queries are allowed. The final answer does not count toward this restriction.

Example

Program output	Grader output
	5
QUERY 10001	
	0 0 1 2 0
QUERY 00010	
	1 1 0 0 1
QUERY 10000	
	0 0 0 1 0
ANSWER	
1 4	
4 2	
5 4	
3 5	

Comment

The tree in question is the following one:

```
1-4-2
  |
  5-3
```

With the three queries in the example, it is possible to reconstruct it uniquely.

J – Jumbled Stacks

Time limit: 1 s *Memory limit: 256 MiB*

We are given a set of n cards, labelled from 1 to n, which are distributed into k stacks $S_1, S_2, \ldots, S_k$. Each stack has a limited capacity: the i-th stack, S_i, can contain at most C_i cards. The only way we can manipulate these cards is by taking the top card of a stack and moving it to the top of some other stack (as long as this wouldn't exceed the capacity of the destination stack).

Using a sequence of such moves, we would like to rearrange the cards so that the first few stacks (0 or more) with the smallest indices are filled to capacity, the stack immediately after them is not filled to capacity (and may even be empty) and all stacks after that are completely empty. Moreover, if we stack together all the stacks from S_1 at the bottom to S_k at the top, the cards should be ordered from smallest to largest, with 1 at the bottom and n at the top.

It is guaranteed that $n \leq \left(\sum_{i=1}^{k} C_i\right) - \max_{1 \leq i \leq k} C_i$.

Suppose we had $n = 6$ cards on $k = 3$ stacks, with capacities $C_1 = 4$, $C_2 = C_3 = 3$, and with the following initial state: $S_1 = [2, 3, 0, 0]$ (from bottom to top; 0 indicates an empty slot), $S_2 = [4, 1, 6]$, $S_3 = [5, 0, 0]$. Then the desired end state is $S_1 = [1, 2, 3, 4]$, $S_2 = [5, 6, 0]$ and $S_3 = [0, 0, 0]$.

Input data

The first line contains two integers, n (the number of cards) and k (the number of stacks), separated by a space. The remaining k lines describe the initial state of the stacks; the i-th of these lines describes S_i and contains $C_i + 1$ integers, separated by spaces. The first of these integers is C_i (the capacity of the stack S_i), the rest of them are the labels of the cards on S_i, from bottom to top. If the stack S_i contains fewer than C_i cards (it could even be empty), the last few integers in the line will be 0.

Input limits

- $1 \leq n \leq 100$
- $3 \leq k \leq 100$
- $1 \leq C_i \leq n$

Output data

Print a sequence of moves that bring the stacks into the desired end state. For each move, output a line containing two integers, separated by a space: first the number of the stack from which the card is being moved and then the number of the stack to which it is being moved (the stacks are numbered from 1 to k; the destination stack must not be the same as the source stack). The number of moves must not exceed 10^5. After the end of the sequence of moves, print a line containing "`0 0`" (without the quotation marks). If there are several possible solutions, you may output any of them.

Example

Input

```
6 3
4 2 3 0 0
3 4 1 6
3 5 0 0
```

Output

```
2 3
2 3
1 2
1 2
3 1
2 1
2 1
3 2
3 1
2 3
1 3
2 1
3 2
3 2
0 0
```

Comment

This is the example discussed earlier in the problem statement. The sample output shows a sequence of 14 moves which bring the stacks into the desired end state.

K – Keys

Time limit: 4 s *Memory limit: 256 MiB*

Alice and Bob live in a massive mansion with n rooms (one of them denoting outdoors, where they play the Moon game) and m doors between them. Each door connects two rooms or a room with the outdoors and has a single unique key that opens this door only. Every door closes behind you and locks automatically, so one always needs a key to pass through. The building is very large, but Alice and Bob only use one room – their bedroom. Other rooms are only there to make the house look bigger and make the neighbors jealous.

This unusual way of building their house is now causing some trouble for Alice and Bob. Bob is leaving for a two-week-long trip. In a week, Alice is also going abroad for a month and when she leaves, she needs the right keys to leave the house. However, Bob also needs keys to get back in since Alice will not be there at the time to let him in. Alice and Bob are now trying to figure out how to split the keys to the doors for Alice to be able to get from room 0 (their bedroom) to 1 (outdoors) and Bob to go from room 1 (outdoors) to 0 (bedroom) one week later.

Fortunately, Alice remembered that she could drop some of the keys on her way out for Bob to pick up on his way back. This way, they can both pass through the same doors. She, of course, cannot drop any keys in room 1 (outdoors) since the neighbors could find them and break into their house.

Can you help Alice and Bob split their keys and plan their trip through the house?

Task

You are given the description of Alice and Bob's mansion: m doors between n rooms numbered 0 to $n-1$, where 1 is outdoors and 0 is the bedroom. The i-th door can be opened with the key number i (0-indexed).

You should first print two lines containing space-separated numbers of keys for Alice and Bob, respectively. It is fine if they do not use all the keys, but it is not allowed for them to both have a copy of the same key (or for one of them to have multiple copies of a key).

You should then print instructions that Alice and Bob will follow. First, print Alice's movements from room 0 to 1 by printing commands of one of the two types:

- "`MOVE x`" to move to room x (assuming that there is a door between Alice's current location and x and that Alice has the key),
- "`DROP k1 k2 ...`" to drop keys $k_1, k_2, \ldots$ (printed as space-separated integers) in the room where Alice currently stands. This means that Alice no longer carries these keys.

Once Alice is done with her movements, print "`DONE`" in a new line. She should finish her movement in room 1. It is fine if Alice passes through room 0 or 1 multiple times while following the printed instructions.

Second, print Bob's movements from room 1 to 0 by printing commands of one of the two types:

- "`MOVE x`" to move to room x (assuming that there is a door between Bob's current location and x and that Bob has the key),
- "`GRAB`" to grab any keys in the room where Bob is currently standing. Bob always grabs all the keys that Alice left in the room. If Alice left none, he does not grab any.

Once Bob is done with his movements, print "`DONE`" in a new line. He should finish his movement in room 0. It is fine if Bob passes through room 0 or 1 multiple times while following the printed instructions.

Remarks: It is considered acceptable, albeit useless, to `DROP` an empty list of keys or to grab keys in a room that has no keys, or grab keys in room 1 (i.e. outside).

Input data

The first line contains integers n and m, the number of rooms (outdoors included) and number of doors. This is followed by m lines that describe the doors. The i-th line (count starting from 0) is described by integers a_i, b_i, indicating that there is a door between rooms a_i and b_i, opened with key i.

Input and output limits

- $2 \leq n, m \leq 10^5$
- $0 \leq a_i, b_i < n$
- It is guaranteed that it's always possible to reach any room from any other room if you have all the keys.
- Each pair of rooms is connected with at most one door.
- No room is connected to itself.
- Your program may print at most $4 \cdot 10^5$ instructions.

Output data

First, print two lines, describing the split of keys. Then, print all the instructions for Alice and Bob, as described in the task description, one per line. If there is no solution, print "`No solution`" (without the quotes). If there are multiple valid solutions, any of them is accepted.

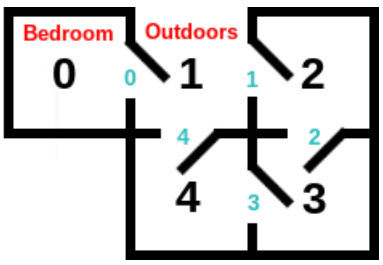
Examples

Input	Output
5 5 0 1 1 2 2 3 3 4 4 1	0 1 2 3 4 MOVE 1 MOVE 2 MOVE 3 DROP 0 MOVE 2 MOVE 1 DONE MOVE 4 MOVE 3 GRAB MOVE 4 MOVE 1 MOVE 0 DONE

Input	Output
3 2 0 2 1 2	No solution

Comment

The first example represents the following floor plan, where blue numbers correspond to key IDs required for each door:



Alice takes keys 0, 1, and 2 while Bob takes keys 3 and 4. Alice walks from 0 to 1, then to 2, and then to 3. There, she drops key 0. She retraces her way back to 1. Bob starts in 1, walks to 4, then to 3, where he picks up key 0. He retraces his way back to 1, and with the newly picked-up key 0, opens up the door to 0.

In the second example, there is no way for both Alice and Bob to reach their destination. Note that Alice cannot drop keys in room 1.

L – Labelled Paths

Time limit: 15 s *Memory limit: 512 MiB*

We are given a directed acyclic graph with n vertices and m edges. Each edge has a *label* (a string of lowercase letters; possibly even an empty string). We can now extend the concept of labels from edges to paths by defining the *label of a path* as the concatenation of the labels of the edges that constitute this path (in the same order in which they appear in the path). The *smallest path* from a start vertex s to a destination vertex t is the path (from s to t) whose label is lexicographically smallest (i.e. the earliest in lexicographical order) amongst all the paths from s to t. Write a program that, for a given s, outputs the smallest paths from s to t for all vertices t of the graph.

Input data

The first line contains four space-separated integers: n (the number of vertices), m (the number of edges), d (the length of the string A, on which see below) and s (the number of the start vertex). The vertices are numbered by integers from 1 to n.

The second line contains a string A, which is exactly d characters long; all these characters are lowercase letters of the English alphabet. All the edge labels in our graph are substrings of the string A.

The remaining m lines describe the edges of the graph. The i-th of these lines describes the i-th edge and contains four space-separated integers: u_i (the start vertex of this edge), v_i (the end vertex of this edge), p_i and ℓ_i. The last two of these integers indicate that the label of this edge is the substring of A that begins with the p_i-th character of A and is ℓ_i characters long. For this purpose we consider the characters of A to be indexed by integers from 1 to d.

Input limits

- $1 \leq s \leq n \leq 600$
- $1 \leq m \leq 2\,000$
- $1 \leq d \leq 10^6$
- $1 \leq u_i \leq n$, $1 \leq v_i \leq n$, $u_i \neq v_i$ (for all $i = 1, \ldots, m$)
- $1 \leq p_i$, $0 \leq \ell_i$, $p_i + \ell_i - 1 \leq d$ (for all $i = 1, \ldots, m$)
- The graph is acyclic and has no parallel edges (i.e. from $i \neq j$ it follows that $u_i \neq u_j$ and/or $v_i \neq v_j$).

Output data

Output n lines, where the t-th line (for $t = 1, \ldots, n$) describes the smallest path from s to t. If there is no path from s to t, the line should contain only the integer `0` and nothing else. Otherwise the line should start with the number of vertices on the path (including vertices s and t), followed by the list of those vertices, separated by spaces. If there are several possible solutions, you may output any of them.

Example

Input

```
5 7 6 3
abcbca
3 2 1 1
2 1 5 1
2 5 4 2
3 1 1 2
3 4 3 2
1 4 6 1
5 4 5 2
```

Output

```
2 3 1
2 3 2
1 3
3 3 1 4
3 3 2 5
```

Comment

In this example, the edge $3 \to 1$ has the label `ab`; the edge $1 \to 4$ has the label `a`; the smallest path from 3 to 4 is $3 \to 1 \to 4$, whose label is `aba`.