ICPC Latin American Regional Contests – 2024

Contest Session: November 9, 2024

Testing Environment and Submission System

1 Information on the Testing Environment

1.1 Environment

The submission correction system will run on the `Ubuntu GNU/Linux 22.04 LTS amd64` distribution, with the following compilers and interpreters configured:

```
C: gcc version 11.4.0 (Ubuntu 11.4.0-1ubuntu1~22.04)
C++20: gcc version 11.4.0 (Ubuntu 11.4.0-1ubuntu1~22.04)
Python: Python 3.10.12
Java: openjdk 17.0.11
Kotlin: kotlin 1.9.10 (JRE 17.0.11+9-Ubuntu-122.04.1)
```

1.2 Memory limits

```
C, C++20, Python: 1GB
Java, Kotlin: 1GB + 100MB stack
```

1.3 Time limits

Before the contest, the judges will have solved all problems in languages from at least two of the three distinct language groups (C/C++, Java/Kotlin, and Python3). Time limits for each problem will be calculated based on the runtime of those solutions.

A link to the document containing time limits for each problem will be available on the `Problems` tab of Boca's web interface.

1.4 Other limits

```
Source file size: 100KB
Output size limit: Specified per problem, alongside the time limits.
```

1.5 Compilation commands

```
C: gcc -x c -g -O2 -std=gnu11 -static -lm
C++20: g++ -x c++ -g -O2 -std=gnu++20 -static
Java: javac
Python: python3 -m py_compile
Kotlin: kotlinc -J-Xms1024m -J-Xmx1024m -J-Xss100m -include-runtime
```

1.6 C/C++20

- Your program must return a zero, executing, as the last command, `return 0` or `exit(0)`.
- It is known that for problems with very large inputs, `iostream` objects can be slow as, by default, they use a buffer synchronized with the stdio library. If you want to use `cin` and `cout`, disabling such synchronization is advised. This can be achieved by calling `std::ios::sync_with_stdio(false);` at the beginning of your main function. Note that, in this case, using `scanf` and `printf` in the same program should be avoided, since, with separate buffers, misbehavior might occur.

1.7 Java

- DO NOT declare a `package` in your Java program; if you declare a package the program will not execute in Boca.
- Notice that the convention for the solution file name must be obeyed, which means that your public class name must be a capital letter (A, B, C, ...).
- Command for running a Java solution: `java -Xms1024m -Xmx1024m -Xss100m {problem_code}`

1.8 Python

- Note that only Python 3 is supported.
- Python programs will be "syntax checked" when submitted; programs which fail the syntax check will receive a "Compilation Error" verdict.
- Please be aware that Python solutions may not be able to meet the time constraints for some problems. While Python is a versatile and user-friendly language, it may not always provide the best runtime performance for certain tasks.

1.9 Kotlin

- DO NOT declare a `package` in your Kotlin program; if you declare a package the program will not execute in Boca.
- Notice that the convention for the solution file name must be obeyed, which means that your source file must be named (A.kt, B.kt, C.kt, ...).
- Command for running a Kotlin solution: `java -Xms1024m -Xmx1024m -Xss100m {problem_code}Kt`

2 Instructions for the Usage of the Boca Submission System

2.1 Submission of Solutions

To submit a solution, you must use the Boca's web interface:

- Open your browser.
- Login as a team (username and password assigned to your team).
- Access the tab `Runs`. Choose the appropriate problem, the language used and upload the source file.

The verdicts you may receive from the judges are:

- 1 - YES
- 2 - NO - Compilation error
- 3 - NO - Runtime error
- 4 - NO - Time limit exceeded
- 5 - NO - Wrong answer
- 6 - NO - Contact staff

Meanings of 1, 2, 3, 4 and 5 are obvious. 6 is used for unforeseeable circumstances. In this case, use the "Clarifications" menu and provide the "run" number for further clarification.

Your program may be run on multiple input files. Note that this means that if your program has more than one error (say, Time Limit Exceeded and Wrong Answer), then you can get either error as verdict.

There is no such thing as "Presentation Error" or "Format Error". If you misspell the word "impossible", for example, and the problem requires that word as output, then your submission will be judged as "Wrong Answer".

Output formatting should follow the sample output in the problem statement, although extra whitespace within reason is acceptable. For example, if you print out thousands of blank spaces within the time and output limit of the problem, that would be judged as a Wrong Answer; however an extra blank at the end of a line or an extra blank line is acceptable.

If your problem outputs more content than the specified `Output size limit`, your submission will receive a Runtime Error verdict. Note that any content written to the standard error stream also counts toward this limit.

Penalties

Each submission to a problem that receives a "`NO`" verdict before the first "`YES`" verdict for the same problem will incur a time penalty of 20 minutes that is added to the total time of the team. There is no penalty time for unsolved problems.

Exceptions for this rule are "`NO - Compilation error`" and "`NO - Contact staff`" that have no time penalty associated with them.

Response Times

Note that the time required for coming up with a verdict varies depending on the problem, the verdict, and the point at which the contest is in when the submission is received. As solutions are judged simultaneously, this means that you might receive the verdict of your runs out of order. Also, in some cases, manual verification is required from the judges. Depending on what needs to be verified and the number of submissions that require manual verification, even delays on the order of minutes are expected.

2.2 Clarifications

All communication with the judges is done through clarification messages.

To request a clarification concerning a problem statement, you must use Boca's web interface:

- Open your browser.
- Login as a team (username and password assigned to your team).
- Access the tab `Clarifications`. Choose the appropriate problem and type your question.

Both clarification replies from the judges and requests sent by you are displayed there. There will be an alert once your clarification is replied (or the judges issue a global clarification).

Note that it's quite uncommon to have clarifications issued and most of the time the answer to the questions received is already on the problem statement. Please read the problem statement and examine the sample test cases carefully before submitting any request for clarifications.

2.3 Score Board

To visualize the score board showing the teams ranking, you must use Boca's web interface:

- Open your browser.
- Login as a team (username and password assigned to your team).
- Access the tab `Score`. You will have access to the local score board.

2.4 Tasks

The tab `Tasks` allows the team to send files for printing, as well as ask for help from a staff member.

- To print a file, just select it from the disk and click on `Send`.
- To ask for help click on the `S.O.S.` button. Note: the help provided by the staff has to be only related to issues with the computers or other physical problem.

ICPC International Collegiate Programming Contest // 2024-2025
The 2024 ICPC Latin America Contests

ICPC Latin American Regional Contests – 2024

November 09, 2024

Contest Session

This problem set contains 12 problems; pages are numbered from 1 to 24.

This problem set is used in simultaneous contests with the following participating countries:

Antigua y Barbuda, Argentina, Bolivia, Brasil, Chile, Colombia, Costa Rica, Cuba, El Salvador, Guatemala, México, Perú, Puerto Rico, República Dominicana, Trinidad y Tobago, Uruguay and Venezuela

v1.0

General information

Unless otherwise stated, the following conditions hold for all problems.

Program name

1. Your solution must be called *codename*`.c`, *codename*`.cpp`, *codename*`.java`, *codename*`.kt`, *codename*`.py3`, where *codename* is the capital letter which identifies the problem.

Input

1. The input must be read from standard input.
2. The input consists of a single test case, which is described using a number of lines that depends on the problem. No extra data appear in the input.
3. When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input. There are no empty lines.
4. The English alphabet is used. There are no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, has the usual end-of-line mark.

Output

1. The output must be written to standard output.
2. The result of the test case must appear in the output using a number of lines that depends on the problem. No extra data should appear in the output.
3. When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output. There should be no empty lines.
4. The English alphabet must be used. There should be no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, must have the usual end-of-line mark.

Problem A – Append and Panic!

Today is Gabriel's first day at his new job, and he has been given his first task. He needs to read a string made up of uppercase letters from a file, sort the letters in the string alphabetically, filter out repeated letters, and then write the result back to the original file. For instance, sorting the string "ICPC" would produce "CCIP", which would become "CIP" after removing repeated letters. Easy, right?

However, Gabriel made a tiny mistake. Instead of overwriting the file with the filtered string, he accidentally appended it to the file. Now, the file is corrupted, containing the original string followed by the sorted, duplicate-free version of it, and Gabriel is in a bit of a panic.

Given the content of the corrupted file, can you determine the length of the original string? Gabriel is confident that with this information, he will be able to complete his assigned task.

Input

The input consists of a single line that contains a string S made up of uppercase letters ($2 \leq |S| \leq 2000$), which is the concatenation of the original (uncorrupted) string t and the sorted, duplicate-free version of t.

Output

Output a single line with an integer indicating the length of t.

Sample input 1	**Sample output 1**
ICPCCIP	4

Sample input 2	**Sample output 2**
ABEDCCCABCDE	7

Sample input 3	**Sample output 3**
ZZ	1

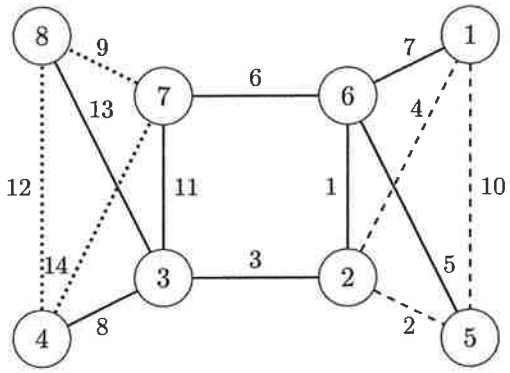
This page would be intentionally left blank if we would not wish to inform about that.

Problem B – Biketopia's Cyclic Track

Biketopia is a country proud of its thriving bike culture. There are N cities and M two-way roads in the country. Each road is fully prepared for cyclists and connects a different pair of cities, in such a way that it is possible to travel between any two cities using one or more roads. Besides, each city is an endpoint of at least three roads.

With the annual bicycle racing tournament approaching, Biketopia needs to select a cyclic track along the roads to serve as the competition route. This track must form a closed loop of roads, passing through at least three distinct cities. While the track can pass through the same city multiple times, no road can be used more than once.

On race day, the roads selected for the track will be closed to the public, but the cities on the track will remain accessible. In past tournaments, some cities became unreachable due to road closures, causing significant disruptions. This is where your help is needed. Your task is to find a cyclic track such that, even after the track's roads are closed, it remains possible to travel between any two cities in the country.



As an example, consider the illustration above, which corresponds to the first sample. The roads highlighted with dotted lines (14, 9 and 12) indicate the chosen competition track. This track forms a closed loop, visiting three different cities (4, 7 and 8). Note that it remains possible to travel between any pair of cities using the remaining roads. Additionally, another possible track is shown with dashed lines, illustrating that the solution is not unique.

Input

The first line contains two integers N ($4 \leq N \leq 2 \cdot 10^5$) and M ($6 \leq M \leq 3 \cdot 10^5$), indicating respectively the number of cities and the number of two-way roads. Each city is identified by a distinct integer from 1 to N, and each road is identified by a distinct integer from 1 to M.

The i-th of the next M lines describes road i with two integers U and V ($1 \leq U, V \leq N$ and $U \neq V$), indicating that the endpoints of the road are cities U and V.

It is guaranteed that there is at most one road between each pair of cities, each city is an endpoint of at least three roads, and it is possible to travel between any two cities using one or more roads.

Output

If a valid track exists, output two lines. The first line must contain the number of roads in the track, and the second line must contain the identifiers of these roads listed in the order they appear in the track. If there are multiple solutions, output any of them.

If no valid track exists, output a line with the character "*" (asterisk) instead.

Sample input 1	Sample output 1
8 14 2 6 2 5 2 3 2 1 6 5 6 7 6 1 3 4 7 8 1 5 3 7 4 8 3 8 4 7	3 14 9 12

Problem C – Cindy's Christmas Challenge

The Christmas season is approaching! The town of Who-ville is getting colorful again, and so are the houses of its citizens, the Whos.

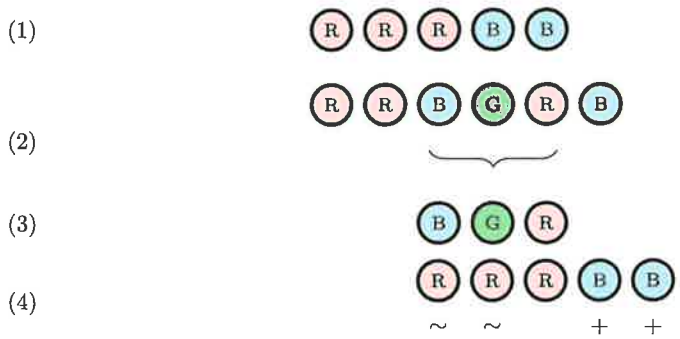
Cindy Lou Who, a young girl living in the city, has a very peculiar Christmas tradition: Cindy goes through her neighborhood, door-to-door, asking for red and blue Christmas tree balls. She gathers all the balls she collects and arranges them in her backyard. This Christmas, Cindy collected R red balls and B blue balls, which she arranged in a sequence with all red balls first, followed by all blue balls. Cindy called this arrangement an (R, B)-sequence.

Near Who-ville lives Grinch, a cranky and solitary creature who hates Christmas and envies the Whos' holiday cheer. He sneaked into Cindy's backyard at night and replaced her (R, B)-sequence with an arbitrary sequence S of red, blue, and green balls (which isn't a color Cindy likes, as it doesn't contrast well with Christmas trees).

The Whos, known for their cheerful Christmas spirit, immediately offered to help Cindy recover her (R, B)-sequence. Although she was initially mad at whoever messed in her backyard, she came up with an idea to turn this into a joyful Christmas game.

The game works as follows. For each citizen, Cindy will pick a contiguous subsequence, $S_L, S_{L+1}, \ldots, S_U$, from Grinch's sequence and ask the citizen for the minimum number of operations needed to transform the subsequence into an (R, B)-sequence. An operation consists of adding, removing or replacing a ball at any position within the subsequence. Note that each subsequence is not actually transformed into an (R, B)-sequence; the citizen must just inform the minimum number of operations required.

Your task is to compute, for each Who, the minimum number of operations required for their assigned contiguous subsequence.



As an example, for $R = 3$ and $B = 2$, consider the picture above. At the top of the picture is Cindy's original $(3, 2)$-sequence (1). Below that is the sequence S after Grinch's attack (2). A possible contiguous subsequence with $L = 3$ and $U = 5$ follows (3). Finally, a way to transform the subsequence into a $(3, 2)$-sequence using four operations is shown, replacing the first two balls with red balls and adding two blue balls at the end (4). Four is the minimum number of operations required for this contiguous subsequence.

Input

The first line contains two integers R and B ($1 \leq R, B \leq 10^6$), indicating respectively the number of red balls and the number of blue balls in Cindy's (R, B)-sequence.

The second line contains a string S ($1 \leq |S| \leq 5 \cdot 10^5$) describing Grinch's sequence. Each character in S is one of the uppercase letters "R", "B" or "G", indicating respectively a red, blue or green ball.

The third line contains an integer W ($1 \leq W \leq 10^5$) representing the number of citizens in Who-ville.

Each of the next W lines describes a contiguous subsequence of S with two integers L and U ($1 \leq L \leq U \leq |S|$), indicating that the subsequence $S_L, S_{L+1}, \ldots, S_U$ is assigned to a citizen.

Output

For each contiguous subsequence described in the input, output a line with an integer indicating the minimum number of operations needed to transform the subsequence into an (R, B)-sequence. Output the results in the same order that the corresponding subsequences appear in the input.

Sample input 1	**Sample output 1**
3 2 RRBGRB 3 3 5 1 5 1 3	4 3 2

Sample input 2	**Sample output 2**
2 3 RRGRBR 6 1 3 1 2 1 4 3 3 3 5 5 6	3 3 3 5 3 4

Sample input 3	**Sample output 3**
5 6 GRBBBB 4 4 5 1 4 5 6 2 2	9 8 9 10

Sample input 4	**Sample output 4**
1 1 RB 2 1 1 1 2	1 0

Problem D – Diverse T-Shirts

The Innovative Clothing Pattern Championship (ICPC) is approaching, and the organizers are determined to ensure that the selection of contestants' T-shirts meets the competition's high standards of variety and style.

To achieve this, they are negotiating with a T-shirt vendor that offers N distinct models. Each model is defined by a unique combination of a text color and a background color, and is assigned an integer from 1 to N to identify this combination.

To keep the selection visually diverse, the organizers have decided to follow a strict diversity rule: no two selected models can share the same text color or the same background color. For example, if one model has "red text on a blue background" and another has "red text on a white background", only one of them can be chosen because they share the same text color. Conversely, models like "white text on a black background" and "black text on a white background" do not share either the text or background color, so both may be chosen.

To assist the organizers, the vendor has prepared an $N \times N$ binary matrix A that identifies all incompatible pairs of models. Specifically:

- $A_{i,j} = 1$ if $i \neq j$ and models i and j cannot both be chosen because they share either the text color or the background color, and
- $A_{i,j} = 0$ otherwise.

The organizers need your help to determine if it is possible to select enough T-shirt models for the event while respecting the diversity rule. Using the vendor's matrix, can you determine the maximum number of models that can be selected?

Input

The first line contains an integer N ($1 \leq N \leq 1000$) indicating the number of T-shirt models.

Each of the next N lines contains a binary string of length N. In the i-th string, the j-th character indicates the value of $A_{i,j}$ ($i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, N$).

It is guaranteed that at least one set of T-shirt models corresponding to the input matrix exists.

Output

Output a single line with an integer indicating the maximum number of T-shirt models that can be chosen without violating the diversity rule.

Sample input 1	**Sample output 1**
3 011 101 110	1

Sample input 2	**Sample output 2**
3 010 101 010	2

This page would be intentionally left blank if we would not wish to inform about that.

Problem E – Evereth Expedition

After an unforgettable trip through Nlogonia's scenic viewpoints, Lola and her husband are back to tackle their most ambitious adventure yet: the Evereth Expedition. This famous mountain has N stations built at distinct altitudes, each offering breathtaking views and a chance to rest along the way. Stations are identified by integers from 1 to N, ordered by altitude (from lowest to highest). Multiple trails connect the stations and the base of the mountain, ensuring it is safe to move between the base and any station, as well as between any pair of stations.

To make the most of their hike, Lola has carefully planned an itinerary that, starting from the base of the mountain, would take the couple through each station exactly once. Since going up and down multiple times would require extra effort, the itinerary consists of a single ascent to the peak and then a descent back to the base. However, just before Lola could double-check the itinerary for any mistakes, she accidentally spilled coffee on her notes, leaving some station numbers unreadable. Without time to plan it all over again, Lola needs your help.

Your task is to complete the itinerary by filling in the unreadable numbers, so that it is possible to visit each station exactly once, in a single ascent followed by a single descent. Note that it is also valid to visit all the stations on the way up or all on the way down.

Input

The first line contains an integer N ($1 \leq N \leq 10^5$) indicating the number of stations. Each station is identified by a distinct integer from 1 to N.

The second line contains N integers $A_1, A_2, \ldots, A_N$ ($0 \leq A_i \leq N$ for $i = 1, 2, \ldots, N$), representing the planned itinerary, where $A_i = 0$ indicates an unreadable station. It is guaranteed that all readable stations are unique, that is, $A_i \neq A_j$ or $A_i = A_j = 0$ for each $i \neq j$. Additionally, there is at least one unreadable station, meaning there exists i such that $A_i = 0$.

Output

Output a single line with N integers describing any itinerary that visits each station exactly once, in a single ascent followed by a single descent. Besides, the itinerary must match the input itinerary in all but the unreadable stations.

If such an itinerary does not exist, output a line with the character "*" (asterisk) instead.

Sample input 1	**Sample output 1**
5 3 0 5 0 0	3 4 5 2 1

Sample input 2	**Sample output 2**
5 4 2 0 0 1	*

Sample input 3	**Sample output 3**
5 0 0 0 0 0	5 4 3 2 1

This page would be intentionally left blank if we would not wish to inform about that.

Problem F – Finding Privacy

You surely agree that the best place to use a restroom is at home. However, sometimes people have no choice but to use a public restroom, where toilets are often arranged side by side in a single row. Aiming for some privacy, each person who enters such restroom will choose an unoccupied toilet that has no occupied toilets on its sides.

Suppose that K people arrive at a public restroom with N initially unoccupied toilets arranged in a row. Determine if it's possible that each of the K people chooses a toilet with no occupied toilets on its sides, and an additional person would not be able to find an unoccupied toilet meeting this privacy condition. People choose toilets one by one, and each chosen toilet is immediately occupied before the next person is allowed to choose.

Input

The input consists of a single line that contains two integers K and N ($1 \leq K \leq N \leq 10^6$), indicating respectively the number of people and the number of toilets.

Output

Output a single line with a string of length N if the K people can choose toilets in a way that prevents an additional person from finding an available toilet with the required privacy. In this case the i-th character of the string must be the uppercase letter "X" if the i-th toilet is chosen, and the character "-" (hyphen) otherwise. Toilets are chosen as it is described in the statement. If there are multiple solutions, output any of them.

If toilets cannot be chosen as requested, output a line with the character "*" (asterisk) instead.

Sample input 1	**Sample output 1**
1 5	*

Sample input 2	**Sample output 2**
2 5	-X-X-

Sample input 3	**Sample output 3**
3 5	X-X-X

Sample input 4	**Sample output 4**
4 5	*

Sample input 5	**Sample output 5**
5 5	*

Sample input 6	**Sample output 6**
2 5	-X--X

This page would be intentionally left blank if we would not wish to inform about that.

Problem G – Grand Glory Race

The Kingdom of Arboria has developed a vast network of N villages numbered from 1 to N. These villages are interconnected by $N-1$ two-way roads, forming an unrooted tree.

These days, the kingdom's streets are buzzing with excitement over the upcoming Grand Glory Race. This prestigious race has been losing popularity in recent years. Over time, all the runners became so well-prepared that now they all run at the same speed. Because all runners move at the same pace, the order in which they reach each village is determined purely by the starting point, as each runner always takes a shortest path toward the finish line. Thus, it became easy to predict who would win the race, taking away much of the interest.

But this year, the race organizers have introduced some intriguing new rules to make the competition more engaging:

- **The Crown Village is Still a Mystery**: The race will conclude at the *Crown Village*, but no one knows yet which village it will be. The suspense is building with speculations, keeping everyone guessing and adding excitement to the event.
- **The Racers**: Each *leaf village* (a village with only one connecting road) will send one runner to participate. These runners will start from their respective leaf villages and race toward the Crown Village along a shortest path. Once a runner reaches the Crown Village, the race ends for them, but other runners continue their journey toward the Crown Village until they arrive.
- **Claiming Villages**: To make the race more appealing for every village, the first runner to reach a village claims it. If multiple runners arrive at a village at the same time, the runner starting from the leaf village with the smaller identifier claims the village. This way, even if a runner doesn't reach the Crown Village first, they can still claim more villages along the way, keeping the race competitive and unpredictable.

With these new rules in place, bookmakers across the kingdom are analyzing the odds for different runners. While the order in which the runners reach the Crown Village is easy to determine, what really interests the bookmakers is how many villages each runner can claim during their journey.

To assist the King's bookmaker, you are tasked with answering queries about various race scenarios. For each query, you'll be given two numbers: the identifiers of a leaf village S and a village T, hypothetically chosen as the Crown Village. Based on this, you need to determine how many villages will be claimed by a runner starting at S if the race ends at T.

Input

The first line contains an integer N ($2 \leq N \leq 10^5$) indicating the number of villages in the kingdom. Each village is identified by a distinct integer from 1 to N.

Each of the next $N-1$ lines contains three integers U, V and L ($1 \leq U, V \leq N$, $U \neq V$ and $1 \leq L \leq 10^4$), representing that there is a two-way road connecting villages U and V, with length L.

The next line contains an integer Q ($1 \leq Q \leq 10^5$) denoting the number of queries.

Each of the next Q lines describes a query with two integers S and T ($1 \leq S, T \leq N$), indicating respectively the leaf village where a specific runners starts, and the hypothetical Crown Village (the finish line for this particular query).

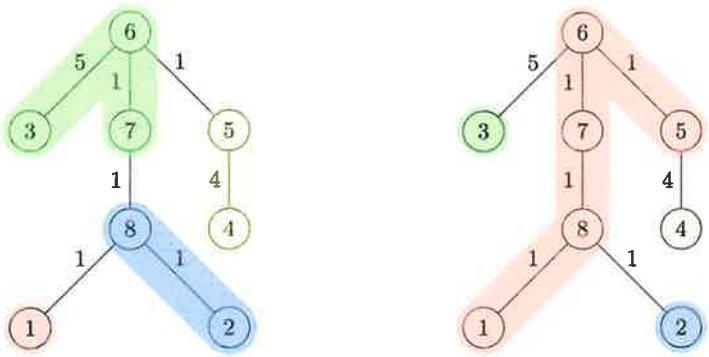
It is guaranteed that the villages and roads form a tree, and that S is a leaf in the tree for each query.

Output

For each query in the input, output a line with an integer representing the number of villages that will be claimed by the runner if the specified Crown Village is chosen as the finish line. Output the results in the same order that the corresponding queries appear in the input.

Sample input 1	Sample output 1
8 6 3 5 6 7 1 6 5 1 5 4 4 7 8 1 8 1 1 8 2 1 8 3 1 1 5 4 5 1 1 2 5 2 1 3 5 4 1	3 5 1 1 1 2 1 2

Explanation of sample 1:
The picture below shows the villages that each runner will claim if the end of the race is at village 1 (left) and 5 (right).



Problem H – Heraclosures

Prometheus is a programmer. He wrote a large program in Heraclosures, a programming language invented by his friend Heracles. The program contains N functions, each identified uniquely by an integer from 1 to N. Each function is made up of many instructions, some of which are call instructions that invoke other functions.

When a call instruction in function i calls a target function j, the full code of function j is executed before continuing with the execution of function i. Each function may contain multiple call instructions, and it is possible for a function to call another function several times. Some functions, however, may have no call instructions at all. Following advice from his friend Sisyphus, Prometheus ensured that there are no cycles among the function calls, avoiding endless call loops.

Each function i has a base execution time B_i, which is its execution time excluding any functions it calls. Execution times are measured in processor cycles, so B_i is always a non-negative integer. The total execution time of function i, denoted as $T(i)$, includes both its base execution time and the total execution time of all functions it directly calls. Formally,

$$T(i) = B_i + \sum_{j \in C(i)} T(j) \ ,$$

where B_i is the base execution time we already defined, and $C(i)$ is the multiset of functions that function i directly calls. If function i calls function j multiple times, then $T(j)$ appears multiple times in this sum. If function i has no call instructions, then $C(i)$ is empty, and so $T(i) = B_i$.

Prometheus wants to assess the impact of several code adjustments on the performance of his program. Tired of programming, he has hired you to implement a tool that accepts a list of events, where each event is either an update or a query.

- An update sets B_i to a given value V. Notice that while this event may modify the base execution time of function i, the multiset $C(i)$ of its called functions remains the same.
- A query calculates the current total execution time $T(i)$ of function i, taking into account all updates that have occurred up to that point in the event list.

As a first approach, Prometheus does not want the result of each query but a single summary result. Suppose there are q queries, numbered sequentially from 1 to q in the list of events. Then, the summary result is calculated as:

$$\left(\sum_{k=1}^{q} k \cdot T(i_k) \right) \bmod (10^9 + 7) \ ,$$

where $T(i_k)$ is the total execution time of the function specified in the k-th query, taking into account all updates up to that query.

Input

The first line contains an integer N ($1 \leq N \leq 8000$) indicating the number of functions. Each function is identified by a distinct integer from 1 to N.

The next line contains N integers $B_1, B_2, \ldots, B_N$ ($0 \leq B_i \leq 10^9$), representing that the base execution time of function i is B_i.

The next line contains an integer M ($1 \leq M \leq 8000$) denoting the number of call instructions among all the functions.

Each of the next M lines describes a call instruction with two integers F and G ($1 \leq F, G \leq N$ and $F \neq G$), indicating that function F contains a call instruction to function G.

The next line contains an integer E ($1 \leq E \leq 10^6$) representing the number of events in the list that your tool must accept.

Each of the next E lines describes an event, in the order they appear in the list.

If the event is an update, the line contains the uppercase letter "`U`" followed by two integers I ($1 \leq I \leq N$) and V ($0 \leq V \leq 10^9$), denoting that the base execution time of function I must be set to V.

If the event is a query, the line contains the uppercase letter "`Q`" followed by an integer J ($1 \leq J \leq N$), indicating that the total execution time of function J must be determined, taking into account all previous updates that appear in the list.

It is guaranteed that there are no cycles among the function calls, and there is at least one query in the list of events.

Output

Output a single line with an integer indicating the summary result of all the queries.

Sample input 1	**Sample output 1**
3 10 20 100 3 1 2 2 3 1 3 3 Q 1 Q 2 Q 3	770

Explanation of sample 1:
Total execution times for each query are: 230, 120 and 100.

Sample input 2	**Sample output 2**
2 42 10 2 2 1 2 1 5 Q 2 Q 1 U 2 0 Q 1 Q 2	640

Explanation of sample 2:
Total execution times for each query are: 94, 42, 42 and 84.

Problem I – Inversion Insight

In MathemIsland, the wildlife is very diverse. There are roots, trees, leaves, pigeons – everything you'd find in a math book. And everywhere you look, there is a permutation.

ICPC University has devised a systematic way to catalog these permutations. Specifically, because inversions are of utmost importance in studying wildlife genetics, ICPC University has decided to sort all $N!$ permutations of the integers from 1 to N: first by the number of inversions and, in the case of a tie, by lexicographic order. This approach uniquely identifies each permutation by an integer from 1 to $N!$, indicating its position in the sorted list of all $N!$ permutations.

Thus, the identity permutation $(1, 2, \ldots, N)$, which is the only permutation with zero inversions, is assigned the identifier 1, while the reverse identity permutation $(N, N-1, \ldots, 1)$, which is the only one with the maximum number of inversions, is assigned the identifier $N!$.

As part of the team implementing the ICPC University database, your task is to retrieve a specific permutation based on its identifier. Write a program that, given two integers N and K, outputs the permutation of the integers from 1 to N corresponding to identifier K.

Remember that the number of inversions in a permutation is the number of pairs of elements that are out of their natural order. That is, for a permutation π with N elements, its number of inversions $\text{inv}(\pi)$ is defined as

$$\text{inv}(\pi) = |\{(i, j) : 1 \leq i < j \leq N \wedge \pi(i) > \pi(j)\}|$$

Input

The input consists of a single line that contains two integers N ($1 \leq N \leq 2 \cdot 10^5$) and K ($1 \leq K \leq \min(N!, 4 \cdot 10^{18})$).

Output

Output a single line with N integers, describing the K-th permutation of the integers from 1 to N, considering permutations sorted according to the university's criteria.

Sample input 1	**Sample output 1**
4 10	1 4 3 2

Sample input 2	**Sample output 2**
5 120	5 4 3 2 1

Sample input 3	**Sample output 3**
16 12345678901234	2 13 8 10 3 15 16 5 11 12 1 9 7 6 14 4

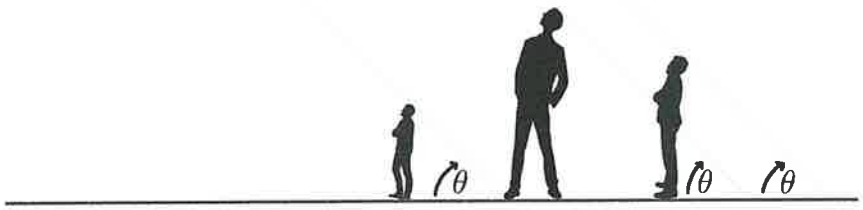
This page would be intentionally left blank if we would not wish to inform about that.

Problem J – Jigsaw of Shadows

They finally did it! The flat-earthers managed to teleport themselves to an idealized flat world where no one can make fun of them anymore.

In this world, there is a perfectly straight, infinite road that runs along the x-axis. Instead of a sun, they have something more illuminating: a gigantic flashlight elevated above the road, positioned infinitely far away to the west (negative x-axis). This flashlight beams light at a precise angle with respect to the ground, illuminating the entire road.

There are N flatlanders standing proudly at distinct positions along the road. As the light strikes each of them, it casts a shadow extending eastward (toward the positive x-axis).



Eager to show off their flat-world knowledge, the citizens want to calculate how much of the road is covered by their shadows. However, with shadows potentially overlapping, they need your help to figure it out. Given the positions of the flatlanders and their heights, can you calculate the total length of the road covered by their shadows?

Input

The first line contains two integers θ ($10 \leq \theta \leq 80$) and N ($1 \leq N \leq 10^5$), indicating respectively the angle of the light beams and the number of flatlanders on the road. The angle is measured in degrees, clockwise from the ground to the light beams.

Each of the next N lines contains two integers X ($0 \leq X \leq 3 \cdot 10^5$) and H ($1 \leq H \leq 1000$), indicating that a flatlander of height H is located at position X along the road. It is guaranteed that no two flatlanders share the same location.

Output

Output a single line with the total length of the road covered by the shadows of all flatlanders. The output must have an absolute or relative error of at most 10^{-4}.

Sample input 1	**Sample output 1**
45 3 50 150 0 100 100 200	300

Sample input 2	Sample output 2
60 3 50 150 0 100 100 200	215.47

Sample input 3	Sample output 3
30 3 50 150 0 100 100 200	446.41016

Problem K – Kool Strings

Professor Kardashi is known for always being fashionable and for her passion for computer science. Her current obsessions are binary strings and efficiency. In particular, she says that a binary string is *kool* if it does not contain K or more consecutive identical characters.

To test your skills, Professor Kardashi gives you a binary string S and allows you to perform the following operation on it: choose an index i and flip the value of S_i (changing a "0" to "1" or a "1" to "0").

Your task is to transform S into a kool string using the minimum number of operations.

Input

The input consists of a single line that contains an integer K and a binary string S ($2 \leq K \leq |S| \leq 10^5$).

Output

Output a single line with an integer indicating the minimum number of operations needed to transform S into a kool string, followed by a kool string that can be obtained after applying that number of operations to S. If there are multiple solutions, output any of them.

Sample input 1	**Sample output 1**
2 00	1 01

Sample input 2	**Sample output 2**
2 10	0 10

Sample input 3	**Sample output 3**
3 1111100	1 1101100

Sample input 4	**Sample output 4**
3 00001111	2 01001101

This page would be intentionally left blank if we would not wish to inform about that.

Problem L – Latin Squares

A Latin square of size N is an $N \times N$ matrix where each entry is an integer between 1 and N, and no two entries in the same row or column are equal.

Alice has a Latin square of size N, and she applied a list of transformations to it. Each transformation swaps either two distinct rows or two distinct columns of the matrix. The transformations are applied sequentially: the first transformation is applied to the original matrix, the second to the result of the first transformation, and so on. The final result is, of course, an $N \times N$ matrix.

Alice will not reveal any matrix to you, only the list of transformations. Your task is to determine whether there exists an original matrix that, after applying all transformations, is identical to the final matrix. If such a matrix exists, you must also provide a possible content for this original matrix.

Input

The first line contains two integers N ($2 \leq N \leq 500$) and T ($1 \leq T \leq 10^5$), indicating respectively the size of the Latin square and the number of transformations.

The next T lines describe the transformations, in the order they are applied, one transformation per line. Each of these lines contains a character X and two integers I and J ($1 \leq I, J \leq N$ and $I \neq J$). The character X is either the uppercase letter "R" or the uppercase letter "C" indicating respectively that rows I and J, or columns I and J are swapped.

Output

If there exists an original matrix that, after applying all transformations, is identical to the final matrix, output N lines with N integers each, representing any such matrix.

If no such matrix exists, output a line with the character "*" (asterisk) instead.

Sample input 1	Sample output 1
4 4 R 1 2 C 2 1 R 3 4 C 3 4	1 2 3 4 2 1 4 3 3 4 1 2 4 3 2 1

Sample input 2	Sample output 2
4 3 R 1 2 R 1 2 R 2 1	*

This page would be intentionally left blank if we would not wish to inform about that.