Fase Zero da Maratona SBC de Programação 2025

May 24 2025

Problem set

General information

This problem set contains 13 problems, the pages are numbered from 1 to 25, including this cover page. Please check if the problem set is complete.

Program names

- For solutions in C, C++, and Python, the source filename is not significant; it can be any name.
- If the solution is in Java or Kotlin, the source file must be named `A.java` or `A.kt` respectively, where `A` should be replaced by the uppercase letter that identifies the problem. Remember that in Java, the name of the main class must be the same as the name of the file.

Input

- The input must be read from the *standard input*.
- The input consists of exactly one single test case, described in a number of lines that depends on the problem. The input format is **exactly** as described in each problem, with no extra content.
- All lines of the input, including the last one, end with the newline character (\n).
- The input does not contain empty lines.
- When the input contains multiple values separated by spaces, there is exactly one whitespace between two consecutive values on the same line.

Output

- The output must be written to the *standard output*.
- The output must adhere to the format specified in the statement without any extra data.
- Whitespace and line breaks are treated equally for grading purposes, and extra spaces are ignored.
- When an output value is a real number, use the number of decimal places corresponding to the precision required in the statement.

Competition rules

- The use of up to three computers for system access is allowed, one for each competitor.
- Competitors do not need to be in the same physical space.
- It is allowed to view and even copy any material (printed, digital, or even online) that was available before the competition started.
- The use of generative AI tools in general, such as ChatGPT, DeepSeek, Gemini, Claude, Copilot, Cursor, among others, is **prohibited**.
- It is allowed for the three competitors of the same team to communicate with each other through a communication channel restricted only to the team.
- Competitors are not allowed to leak any information about the contest to anyone outside their own team.
- Communication between teams is prohibited.
- Communication between competitors and coaches during the contest is not allowed. Coaches cannot participate in the contest, cannot see the problems during the contest, nor can they log into the special account created in BOCA for the competitors.
- Every code submission during the contest must be a legitimate attempt to solve the problem. Attempts to mine test cases and circumvent the system will be penalized.

If behavior that violates any of the above items is identified or if there are reports to be investigated and proven, the offending teams will be disqualified.

Testing environment information

- The time limit can be found in the statement of each problem.
- The memory limit for all problems is 1 GiB.
- The execution stack has a limit of 100 MiB.
- The source file can be a maximum of 100 KiB.
- The output can be a maximum of 1 MiB.

Compilation commands used

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

C/C++

- Your program must return zero, executing, as the last command, `return 0` or `exit(0)`.
- It is known that in some cases of problems with very large inputs, `cin` objects can be slow due to the synchronization of stdio buffers. If you wish to use `cin` and `cout`, one way to work around this problem is to execute the command `ios_base::sync_with_stdio(0)` at the beginning of your `main` function. Note that in this case, using `scanf` and `printf` in the same program is discouraged, as separate buffers can lead to inappropriate behaviors.

Java

- Do not declare "`package`" in your source code.
- The convention of naming the file according to the problem letter must be followed. For example, for problem A, your file must be named `A.java` with a public class `A`.
- Execution command: `java -Xms1024m -Xmx1024m -Xss20m <fileName>`.
- Attention: it is not guaranteed that Java solutions will execute within the allocated time limit.

Kotlin

- Do not declare "`package`" in your source code.
- The convention of naming the file according to the problem letter must be followed. For example, for problem A, your file must be named `A.kt` without declaring the main class (resulting in an implicit class called `AKt`).
- Execution command: `java -Xms1024m -Xmx1024m -Xss20m <fileName>`.
- Attention: it is not guaranteed that Kotlin solutions will execute within the allocated time limit.

Python

- The Python interpreter used is PyPy, which offers more speed in execution.
- Python is at version 3.8.13, and in case of a syntax error, a Runtime Error will be returned.
- Attention: it is not guaranteed that Python solutions will execute within the allocated time limit.

Instructions for using the BOCA submission system

Everything in BOCA is done through the web interface, so before taking any action, make sure that the browser is open on the BOCA judge's page, and that you are logged in.

Using the system

- BOCA does not have dynamic updates. To get the latest updates, ensure that the page content is refreshed (for example, by using F5).
- It is possible to download the statements of the contest and each of the problems in the `Problems` tab.
- To submit solutions and see the results of your submissions, use the `Runs` tab. To make a submission, choose the appropriate problem, the programming language used, and submit the source file.
- To see global clarifications and request clarifications about the statement of a problem, use the `Clarifications` tab. Choose the appropriate problem, write your question, submit it, and wait for the response that will appear on this screen.
- To see the scoreboard, use the `Score` tab.
- The `Tasks` tab is used to submit printing tasks and request assistance in physical locations. In remote competitions, this feature is not used.
- The `Backups` tab provides a file-sharing environment for the team. If it is necessary to switch machines during the competition, use this area to transfer files to the new computer.

Submission result

- The code will be judged by an automatic judge that will evaluate the program's behavior for a set of secret test cases. Although each problem contains example test cases with which competitors can test their codes locally, it is emphasized that the set of secret test cases from the judge is generally much larger. Note that your submission may also be manually reviewed by the competition judges, extending the judging time.
- Each submission can result in one of the following verdicts:
  - YES: your code was accepted, the output of your solution matches that of the judges, and your team won a balloon! Congratulations!
  - NO - Wrong answer: your code was not accepted because the program did not print the expected output for some test case.
  - NO - Time limit exceeded: your code was not accepted because the program took too long to run for some test case.
  - NO - Runtime error: your code was not accepted because the program was aborted by the operating system for performing some invalid operation, such as illegal memory access, excessive memory allocation, or floating-point exception.
  - NO - Compilation error: your code was not accepted because the compiler could not compile it correctly. Check the compilation options used for your programming language. Note that this verdict does not incur a penalty in the score.
  - NO - Name mismatch: only for Java and Kotlin submissions, this verdict is given when the team submits a solution with a main class name different from the specified one.

Problem A

Ambiguous Schrödinger Cat

Time limit: 0.5 s	Memory limit: 1 GiB

In the year 1935, physicist Erwin Schrödinger received a mysterious package labeled only as "Box 42." Inside, a bizarre quantum experiment was set up by a group of eccentric scientists. The box contains a cat, a vial of poison activated by an unstable radioactive atom, and a note beside it with a single instruction: "Do not open unless you are ready to accept a single truth."

According to quantum physics, while the box is closed, it is impossible to determine the state of the cat. The entire system is in a quantum superposition: the cat is both alive and dead simultaneously. However, if someone has the courage to open the box, the superposition collapses, revealing whether the cat is indeed alive or dead.

You, as an apprentice of the old Schrödinger, have found records with the state of the box and the internal reading of the state of the cat; these are privileged information that is not observable to anyone who has not opened the box. Your mission is to determine the observable state of the cat at the described moment.

Input

The only line of input contains two integers C and G ($0 \leq C, G \leq 1$). $C = 1$ indicates that the box is closed, and $C = 0$ indicates that it is open. $G = 1$ indicates that the cat is alive, and $G = 0$ indicates that the cat is dead.

Output

Print a single string, "`vivo e morto`" if it is not possible to know the state of the cat, "`vivo`" if it is possible to determine that the cat is alive, or "`morto`" if it is possible to determine that the cat is dead.

Sample input 1	**Sample output 1**
`1 0`	`vivo e morto`

Sample input 2	**Sample output 2**
`0 1`	`vivo`

Problem B

Periodic Search

Time limit: 0.5 s	Memory limit: 1 GiB

An analysis was conducted on the evolution of the possible quantum states of a system of particles, resulting in a rooted tree of N states. Each state, except for the root state, is connected to its parent state by an edge that has a label of a lowercase Latin letter from `a` to `z`. This label describes the type of interference that caused the system to collapse into another state. The sequence of interferences of a state is defined as the concatenation of the labels of the edges on the path from the root to that state.

For each state, we define the *minimum periodicity* of its sequence as the smallest integer $P \geq 1$ such that the sequence can be obtained by repeating a smaller sequence of length P multiple times (at least twice). If there is no valid integer $P \geq 1$, the sequence is considered to have a periodicity of 0. The empty sequence from the root is considered to have a periodicity of 0.

We are interested in studying periodic interferences within the system of particles, so your task is to determine, among all the states from the root, the highest value of minimum periodicity of their respective interference sequences.

Input

The first line contains an integer N ($2 \leq N \leq 10^5$), the number of states. The second line contains $N-1$ integers $P_1, P_2, ..., P_{N-1}$ ($1 \leq P_i \leq i$), where state $i+1$ is connected to state P_i. The third line contains a sequence of $N-1$ lowercase Latin letters, where the character at position i represents the label on the edge between states P_i and $i+1$.

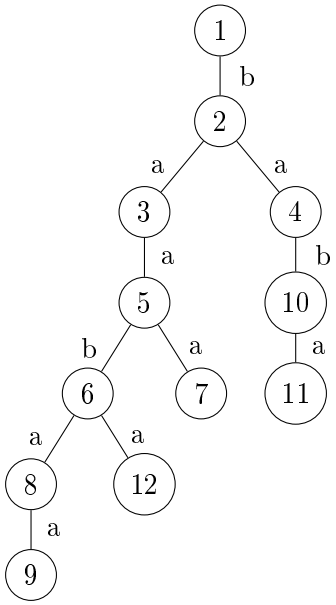
Output

Print a single integer representing the highest minimum period among all sequences formed by the paths from the root to each state.

Sample input 1	**Sample output 1**
12 1 2 2 3 5 5 6 8 4 10 6 baaabaaabaa	3

Explanation of sample 1:

In this example, the described tree is as follows:



We can verify the minimum periodicity in some state sequences of the tree:

- In the sequence that ends at state 2, forming the word `b`, the minimum periodicity is 0 since it is not possible to form this word with two or more repetitions.
- In the sequence that ends at state 11, forming the word `baba`, the minimum periodicity is 2, as it is possible to form this word by repeating the word `ba` twice, which has 2 characters.
- In the sequence that ends at state 9, forming the word `baabaa`, the minimum periodicity is 3, as it is possible to form this word by repeating the word `baa` twice, which has 3 characters.

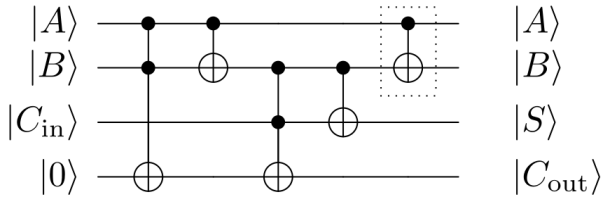
Problem C

Matrix Logic Circuits

Time limit: 0.5 s	Memory limit: 1 GiB

In quantum computing, logic gates operate a bit differently. Quantum logic gates are reversible, and the number of input qubits is equal to the number of output qubits. Additionally, they can be represented by 2^N by 2^N matrices, where N is the number of qubits.

A quantum circuit is a model for quantum computation where the computation is performed through a sequence of quantum logic gates and measurement devices. A sequence of logic gates can be represented by a matrix resulting from the multiplication of the matrices of the logic gates in the order of application, which is the reverse order of how they are graphically represented. For example, the circuit for adding two bits in its quantum form is:



In this circuit, we have two variations of a logic gate that we will call $\text{CNOT}(q_c, q_t)$ and $\text{CCNOT}(q_{c_1}, q_{c_2}, q_t)$. In the diagram, the qubit q_t is marked with a $\oplus$. The logic gate $\text{CNOT}(q_c, q_t)$ can be seen as being equal to $\text{CCNOT}(q_c, q_c, q_t)$, that is, the application of the logic gate CCNOT with $q_c = q_{c_1} = q_{c_2}$.

The logic gate $\text{CCNOT}(q_{c_1}, q_{c_2}, q_t)$ behaves by inverting the output qubit q_t if both control qubits q_{c_1} and q_{c_2} are set. Mathematically, $q'_t = q_t \oplus (q_{c_1} \wedge q_{c_2})$. In its matrix form,

$$\text{CCNOT}(q_{c_1}, q_{c_2}, q_t)_{ij} = \begin{cases} 1 & \text{if } i \text{ has bits } c_1 \text{ and } c_2 \text{ set and } i \oplus 2^t = j \\ 0 & \text{if } i \text{ has bits } c_1 \text{ and } c_2 \text{ set and } i \oplus 2^t \neq j \\ 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

where i is the row and j is the column with $0 \leq i, j < 2^N$, and i contains bit k $(0 \leq k)$ if $\left\lfloor \frac{x}{2^k} \right\rfloor \bmod 2 = 1$. The operation $\oplus$ is the bitwise exclusive or operation, commonly represented as ^ in programming languages.

Thus, the matrix of the quantum circuit for adding two bits is given by

$$\text{CNOT}(q_0, q_1)\,\text{CNOT}(q_1, q_2)\,\text{CCNOT}(q_1, q_2, q_3)\,\text{CNOT}(q_0, q_1)\,\text{CCNOT}(q_0, q_1, q_3),$$

where the qubits q_0, q_1, q_2, q_3 are used with inputs $|A\rangle, |B\rangle, |C_{\text{in}}\rangle, |0\rangle$ respectively and result in $|A\rangle, |B\rangle, |S\rangle, |C_{\text{out}}\rangle$ respectively.

Your task is given the description of a circuit with logic gates CNOT and CCNOT in the order of application, to print the resulting matrix.

Input

The first line of the input contains the integers N $(2 \leq N \leq 8)$, the number of qubits in the circuit, and M $(1 \leq M \leq 10^5)$, the number of logic gates in the circuit.

This is followed by M lines, each with the description of a logic gate. The first integer T ($1 \leq T \leq 2$) defines the type of the logic gate. If $T = 1$, the description is for the logic gate CNOT(q_C, q_T) and is followed by distinct integers C and T ($0 \leq C, T < N$). If $T = 2$, the description is for the logic gate CCNOT(q_{C_1}, q_{C_2}, q_T) and is followed by distinct integers C_1, C_2, and T ($0 \leq C_1, C_2, T < N$). Note that the logic gates are given in the order of application.

Output

Print 2^N lines, each containing exactly 2^N characters '0' or '1', corresponding to the complete matrix of the quantum circuit.

Sample input 1	**Sample output 1**
2 1 1 0 1	1000 0001 0010 0100

Explanation of sample 1:

Here is the translation to English:
This circuit represents only the logical gate CNOT(c, t). If you have read about this logical gate, the matrix may seem incorrect because it is different from what is found in the literature. However, it is a matter of convention. When we compose qubit c with qubit t, we are using the convention that the first bit is less significant than the second.

$$\begin{array}{l} i\text{=0 representing } |00\rangle \text{ with } c\text{=0 and } t\text{=0} \\ i\text{=1 representing } |01\rangle \text{ with } c\text{=1 and } t\text{=0} \\ i\text{=2 representing } |10\rangle \text{ with } c\text{=0 and } t\text{=1} \\ i\text{=3 representing } |11\rangle \text{ with } c\text{=1 and } t\text{=1} \end{array} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

So if the input is $|00\rangle$ or $|10\rangle$, both where $c = 0$ and t varies, the logical gate does not act. When the input is $|01\rangle$ or $|11\rangle$, then the logical gate acts and inverts the value of t. This convention is used, for example, by the Qiskit library.

Sample input 2	**Sample output 2**
4 5 1 0 1 1 1 2 2 1 2 3 1 0 1 2 0 1 3	1000000000000000 0000000000000100 0000000000000010 0000000000010000 0000100000000000 0100000000000000 0010000000000000 0000000000000001 0000000010000000 0000010000000000 0000001000000000 0001000000000000 0000000000001000 0000000001000000 0000000000100000 0000000100000000

Sample input 3	**Sample output 3**
3 1 2 0 1 2	10000000 01000000 00100000 00000001 00001000 00000100 00000010 00010000

Sample input 4	**Sample output 4**
3 1 1 0 1	10000000 00010000 00100000 01000000 00001000 00000001 00000010 00000100

Problem D

Quantum Decoherence

Time limit: 0.5 s	Memory limit: 1 GiB

The SBC (Brazilian Computer Society) is developing various models of architectures for quantum computers, with the goal of making them accessible to everyone in the future. One of the main challenges faced by the development teams is quantum decoherence, which occurs when a qubit in superposition (simultaneously representing states 0 and 1) collapses to 0 or 1 due to environmental interference.

For each model developed, the rate of quantum decoherence will be analyzed. To do this, the qubits will be observed in an isolated state and under normal temperature and pressure conditions. The rate of quantum decoherence is the ratio between the number of qubits that collapsed under normal temperature and pressure conditions and the number of qubits that were in superposition in the isolated state.

Since there are several models, you have been asked to develop a program that calculates this rate. After all, you need extracurricular hours to graduate, right?!

Input

The first line contains an integer N ($10 \leq N \leq 10^5$) indicating the number of qubits in the computer. The next two lines contain the strings S in an isolated state and T under normal temperature and pressure conditions, respectively, of size N, composed of the characters {0, 1, *}, where '*' indicates a qubit in superposition.

It is guaranteed that at least one qubit is in superposition in the isolated state and that every qubit that is not in superposition in string S remains identical in string T.

Output

The output should contain the rate of quantum decoherence in decimal form, with exactly two decimal places.

Sample input 1	**Sample output 1**
10 0*1**100*1 0110*100*1	0.50

Sample input 2	**Sample output 2**
13 *1*01*100*01* 01*0101001011	0.80

Sample input 3	**Sample output 3**
25 *10*1*110*01*011100*110*0 *1011*110001*011100*110*0	0.29

Problem E

Particle Energization

Time limit: 0.5 s	Memory limit: 1 GiB

There is a particle at point $X = 1$ on an infinite number line with a charge value of Y. When interacting with the line, it gains unusual properties: by absorbing energy, this particle releases enough kinetic energy to move $\gcd(X, Y)$ steps along the number line, where $\gcd(X, Y)$ is the greatest common divisor of X and Y. That is, with each procedure, the particle moves from position X to position $X + \gcd(X, Y)$.

Scientists need to energize the particle K times in order to discover new properties about it; however, they need to predict at which point the particle will stop after these procedures so that they can reuse it in future studies.

Therefore, help determine what the final position X will be after the K processes.

Input

The input consists of a line with two numbers Y ($1 \leq Y \leq 10^9$) and K ($1 \leq K \leq 10^9$).

Output

Print an integer containing the position X where the particle will stop following the above procedures.

Sample input 1	**Sample output 1**
`4 3`	`8`

Sample input 2	**Sample output 2**
`7 15`	`70`

Sample input 3	**Sample output 3**
`123 123`	`10086`

Problem F

Feynman Memorizing Numbers

Time limit: 2 s	Memory limit: 1 GiB

Richard Feynman was the first to propose the use of a quantum phenomenon to perform computational routines. This was during a lecture presented at the First Conference on Physics Computing at MIT. He showed that a classical computer would take a long time to simulate a simple quantum physics experiment. Legend has it that he could memorize large sequences of numbers and mentally calculate various properties at super-fast speeds.

The MythBusters, upon learning this, decided to verify this legend directly with Feynman using their time machine. To verify, they would generate a sequence of numbers and ask how many ways we can choose exactly 4 elements from this sequence that sum to X. The creation of the test was assigned to you, the new intern of the MythBusters.

Your task is to write a program that, given a set of numbers and multiple query values, determines how many quadruples $\{i, j, k, l\}$ with $1 \leq i < j < k < l \leq N$ have a sum $A_i + A_j + A_k + A_l$ equal to the queried values.

Input

The input consists of a single test case. The first line contains an integer N ($4 \leq N \leq 1000$), representing the number of numbers in the sequence. The second line contains N integers a_i ($0 \leq |a_i| \leq 1000$), separated by spaces. The third line contains an integer Q ($1 \leq Q \leq 4000$), representing the number of queries. Finally, each of the next Q lines contains an integer q_i ($0 \leq |q_i| \leq 4000$) each, representing the target values queried.

Output

For each query, your program should print a line containing a single integer: the number of quadruples whose sum is equal to q_i.

Sample input 1	**Sample output 1**
8 -1 23 4 -8 4 23 4 5 1 30	6

Explanation of sample 1:

The six quadruples that sum to 30 are:

- $A_1, A_2, A_3, A_7 \rightarrow -1 + 23 + 4 + 4 = 30$
- $A_1, A_3, A_5, A_6 \rightarrow -1 + 4 + 4 + 23 = 30$
- $A_1, A_5, A_6, A_7 \rightarrow -1 + 4 + 23 + 4 = 30$
- $A_1, A_2, A_5, A_7 \rightarrow -1 + 23 + 4 + 4 = 30$
- $A_1, A_3, A_6, A_7 \rightarrow -1 + 4 + 23 + 4 = 30$
- $A_1, A_2, A_3, A_5 \rightarrow -1 + 23 + 4 + 4 = 30$

Problem G

Grover and His Special Paths

Time limit: 1 s	Memory limit: 1 GiB

Grover is an Indo-American computer scientist who wants to propose a quantum search algorithm that offers a quadratic speedup over classical search algorithms for unstructured databases. To achieve this, he needs to solve a problem in trees that is part of his research to develop his algorithm.

Grover has a tree with N vertices and $N-1$ undirected edges, and his goal is to assign a value v_i to each vertex satisfying some constraints:

- $1 \leq v_i \leq 5$
- There are exactly cnt_x vertices with the value $v_i = x$, for $1 \leq x \leq 5$
- For each vertex i, there is a set of values that the value of v_i can take
- Additionally, in this tree, there are P special paths for Grover, where a special path is represented by a pair of vertices (X_i, Y_i) (the paths may intersect, and a path may appear more than once in the input) and for these paths, the values assigned to the vertices along the path (in the order from X_i to Y_i) must form a strictly increasing sequence.

If there is a valid solution, print any one that satisfies the constraints imposed by Grover; otherwise, print -1.

Input

The first line of the input will contain an integer N ($1 \leq N \leq 5 \cdot 10^4$), the number of vertices in the tree. The second line of the input will contain 5 integers, the values of $cnt_1, cnt_2, cnt_3, cnt_4, cnt_5$ respectively. It is guaranteed that the sum $cnt_1 + cnt_2 + cnt_3 + cnt_4 + cnt_5 = N$.

The next N lines of input start with an integer M_i ($1 \leq M_i \leq 5$), followed by M_i distinct integers between 1 and 5, indicating the values that the i-th vertex can take. This is followed by $N-1$ lines, each with two numbers U, V ($1 \leq U, V \leq N$) indicating an edge of the tree.

Next, there will be a line with a single integer P ($1 \leq P \leq 5$), indicating the number of special paths. Finally, the next P lines of input will contain two numbers X_i, Y_i ($1 \leq X_i, Y_i \leq N$), indicating the special paths.

Output

If a solution exists, print N integers v_1, v_2, ..., v_n indicating a valid solution to the problem. Otherwise, print -1. If there are multiple valid answers, any one will be accepted.

Sample input 1	**Sample output 1**
10 1 1 4 2 2 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 1 2 3 4 5 5 6 5 4 5 3 5 8 3 10 10 9 9 1 3 7 10 2 5 7 1 3 10 10 2 5 6 7 6	5 4 2 3 3 5 1 3 4 3

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

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

Sample input 4	**Sample output 4**
10 2 1 4 1 2 5 1 2 3 4 5 4 1 2 3 5 1 3 3 2 4 5 2 4 5 1 3 5 1 2 3 4 5 3 1 2 3 1 4 2 1 5 8 5 5 2 8 9 9 10 8 4 4 1 1 6 9 3 4 7 4 7 10 1 4 6 6 7 10	1 3 3 2 5 3 1 3 4 5

Problem H

Binary Palindromic Harmony

Time limit: 0.5 s	Memory limit: 1 GiB

Several quantum alternatives to traditional algorithms have been researched to solve a variety of problems. Using quantum logic gates, it is possible to gain performance in some specific types of tasks, leading to improvements, for example, in number factorization algorithms and data set searches.

Shor is working on a quantum algorithm for recognizing palindromes in binary sequences. To do this, he needs to generate a large number of examples, and he decided that he would need to know for a given positive integer X, what the largest integer Y less than or equal to X would be, whose binary representation is palindromic.

Shor has a lot of experience with quantum algorithms, but he is having difficulty creating a classical algorithm that solves this problem. Can you help?

Input

The only line of input contains the integer X ($1 \leq X \leq 10^{18}$).

Output

Output a single line with The corresponding integer Y.

Sample input 1	**Sample output 1**
9945	9945

Sample input 2	**Sample output 2**
11	9

Sample input 3	**Sample output 3**
154	153

Problem I

Inspecting the Entanglement

Time limit: 1 s	Memory limit: 1 GiB

In the laboratory of the Institute of Quantum Physics, a team of scientists is conducting a new experiment. The idea is to reconstruct the evolution of a set of entangled particles over time. The problem is that the experiment cannot be observed directly.

Therefore, the laboratory has a network of N quantum sensors distributed in space. The sensors detect various properties of the particles (spin, polarization, angular momentum, etc.). However, for technical and energy reasons, the same sensor cannot be activated for a very short time (to avoid noise) and cannot remain active for too long (to avoid overheating).

The experiment will last T seconds. To avoid wasting energy and to ensure that all information is collected, exactly one of the sensors must be on each second, collecting information, and no sensor should remain on after the end of the experiment.

Each sensor i at time j provides a reliability score $c(i, j)$, reflecting the quality of the measurement at that moment. The team's challenge is to select which sensors to activate throughout the experiment, ensuring that after a sensor is activated, it must remain active for at least L seconds and at most U seconds. Additionally, a sensor can be used multiple times during the experiment, as long as the time it remains active is respected.

The final reliability of the experiment is equal to the sum of the reliability of the chosen sensors at each second during the test. The team's goal is to select the sensors in such a way as to maximize the reliability of the reconstruction of the quantum state. Since the physicists are very busy studying other quantum matters, they have sent this problem to the teams of the Fase Zero da Maratona SBC de Programação to solve.

Input

The first line contains two integers N ($1 \leq N \leq 5 \cdot 10^3$) and T ($1 \leq T \leq 100$) representing the number of sensors and the duration of the experiment, respectively.

The next N lines each contain T integers, where the i-th line's j-th integer will be $c(i, j)$ ($1 \leq c(i, j) \leq 5 \cdot 10^7$).

The last line contains two integers L and U ($1 \leq L \leq U \leq T$) representing the minimum and maximum time that each sensor can remain active.

Output

Output a single line with An integer representing the highest possible reliability. If there is no solution for the given constraints, print -1.

Sample input 1	**Sample output 1**
3 5 2 3 2 1 2 1 1 5 1 2 1 2 2 1 5 1 5	16

Sample input 2	**Sample output 2**
3 5 2 3 2 1 2 1 1 5 1 2 1 2 2 1 5 2 3	13

Sample input 3	**Sample output 3**
3 5 2 3 2 1 2 1 1 5 1 2 1 2 2 1 5 2 2	-1

Problem J

Journey of the Particles

Time limit: 1 s	Memory limit: 1 GiB

An experiment at the Large Hadron Collider is being set up to test phase properties of various different particles. There are N filters arranged in a circle, where the i-th filter allows particles with a phase less than or equal to its limit A_i to pass.

An experiment begins with the release of a particle at the position of the i-th filter, with an initial phase set to A_i. When passing through filter i, including the first one, the particle:

- Is filtered out if the phase of the particle is greater than the limit A_i, ending the experiment.
- Otherwise, the particle passes through the filter, increases its phase by K, and moves to the next filter on the right, the $(i \mod N) + 1$-th filter.

We want to simulate the result of this experiment to compare with the results verified empirically. To achieve this, we need to create a vector B, of size N, where B_i $(1 \leq B_i \leq N)$ is the position of the filter that stopped the particle that started at position i.

Input

The first line contains two integers N $(1 \leq N \leq 2 \cdot 10^5)$ and K $(1 \leq K \leq 10^9)$.

The second line contains N integers $A_1, A_2, \ldots, A_N$ $(-10^9 \leq A_i \leq 10^9)$.

Output

Print a line containing N integers $B_1, B_2, \ldots, B_N$.

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

Sample input 2	**Sample output 2**
5 5 -10 -5 0 5 10	1 1 1 1 1

Sample input 3	**Sample output 3**
6 2 12 4 5 9 2 3	2 3 5 5 6 2

Problem K

K Missing Elements

Time limit: 2 s	Memory limit: 1 GiB

In the core of Station Q-42, scientists have created the most advanced simulator of combinatorial quantum states: the Enhanced Path Entangler, also known as EPE. The EPE has been programmed to study quantum walks in permutation spaces. In a quantum walk, the walker evolves in a superposition of positions, following the rules of quantum mechanics.

The input to the simulator is a permutation A of size N. Each index i of the permutation has an integer A_i associated with it, representing a node in a Hilbert space. We define a quantum walk that follows the increasing evolution criterion as a subsequence of indices $i_1 < i_2 \ldots < i_M$ $(M > 0)$ such that $A_{i_1} < A_{i_2} \ldots < A_{i_M}$. Each node i also has a quantum energy charge stored at position B_i of the vector B.

When simulating all possible quantum walks that follow the increasing evolution criterion, the EPE collapsed the state of each path into a sum of the quantum energies of the visited nodes. Each of these sums was recorded in a vector C, representing all the amplitudes of valid paths collapsed into classical energy.

To organize the data, the vector C was sorted in non-increasing order, but something unexpected happened: a decoherence caused the vector C to get lost in the middle of the simulation. Your goal as an analyst at Station Q-42 is to reconstruct the K largest values of the vector C, or report if C does not have that element.

Input

The first line contains two integers N $(1 \leq N \leq 10^4)$ and K $(1 \leq K \leq 10^5)$.

The second line contains N integers $A_1, A_2, \ldots, A_N$ $(1 \leq A_i \leq N)$, it is guaranteed that A is a permutation.

The third line contains N integers $B_1, B_2, \ldots, B_N$ $(1 \leq B_i \leq 10^4)$.

Output

Print a line containing K integers C_i $(1 \leq i \leq K)$. If a particular C_i does not exist, print -1 in its place.

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

Explanation of sample 1:

The increasing subsequences of A are:

- $\{A_1\} = \{2\} \rightarrow B_1 = 1$
- $\{A_2\} = \{1\} \rightarrow B_2 = 2$
- $\{A_3\} = \{3\} \rightarrow B_3 = 1$
- $\{A_1, A_3\} = \{2, 3\} \rightarrow B_1 + B_3 = 2$
- $\{A_2, A_3\} = \{1, 3\} \rightarrow B_2 + B_3 = 3$

Thus, the vector C after sorting is $\{3, 2, 2, 1, 1\}$. Since $K = 4$, it is only necessary to print $\{C_1, \ldots, C_4\}$.

Sample input 2	**Sample output 2**

`4 16`
`1 2 3 4`
`1 1 1 1`
	`4 3 3 3 3 2 2 2 2 2 2 1 1 1 1 -1`

Explanation of sample 2:

All the $2^4 - 1 = 15$ sequences of A are increasing. Since $K = 16$, then $C_{16} = -1$.

Problem L

qPhones Production Line

Time limit: 0.5 s	Memory limit: 1 GiB

The Brazilian Society of Smartphones (SBC) is developing a new model of smartphones that utilizes quantum computing, the qPhones. Unlike traditional devices that store bits, this new architecture will use qubits.

A qubit (quantum bit) is the basic unit of quantum information, just as a bit is in classical computing. However, while a classical bit can only assume one state at a time (0 or 1), a qubit can exist in superposition, assuming multiple states simultaneously as a "quantum mixture" of 0 and 1, each with a probability of being measured when observed.

Thus, if a device can store qubits, all of its combinations can be represented simultaneously due to superposition. For example, if a device stores 3 qubits, we can have the representation of $2^3 = 8$ superposed classical states, which are 000, 001, 010, 011, 100, 101, 110, and 111.

In practice, we can assume that to simulate 1 qubit we need 2 bits, to simulate 2 qubits we need 4 bits, to simulate 3 qubits we need 8 bits, and so on. Therefore, to fully simulate the memory of a classical cell phone with M megabytes (MB), the engineers at SBC need to ensure that the qubits of the new device can represent at least M megabytes. Consider that 1 MB is equivalent to 10^6 bytes.

You have recently been hired by SBC to assist in the production line of the new quantum smartphones. Your task is, given the memory value of M megabytes, to determine the minimum number of qubits necessary to simulate all possible states of a classical device with that amount of memory.

Input

An integer M ($1 \leq M \leq 10^{10}$) representing the amount of memory in MB of a traditional device.

Output

Output a single line with An integer representing the minimum number of qubits necessary to simulate all possible states of a classical device with M megabytes of memory.

Sample input 1	**Sample output 1**
1	23

Sample input 2	**Sample output 2**
17	28

Problem M

Spooky Movement at a Distance

Time limit: 1 s	Memory limit: 1 GiB

Charles is a great physicist, cryptographer, and computer scientist, known for his significant contributions, including foundational work on the relationship between physics and information. During his studies on quantum teleportation, Charles discovered a quantum field with N positions numbered from 1 to N. In an experiment, Charles can place a particle initially in any position of the quantum field. At each moment in time, the particle can decide whether to teleport to a position greater than its current position or to remain still and finish its path. Thus, there are $2^N - 1$ possible paths.

Let a path be a sequence of positions visited by a particle in an experiment. Each position i ($1 \leq i \leq N$) of the quantum field has an associated coefficient A_i. Charles defines the beauty of a path as the greatest common divisor of all the coefficients of the positions visited in the path.

Charles will perform several operations in sequence, which are:

- Operation `1 X`: Consider that all possible paths have the same probability of being taken. What is the probability that the path taken has beauty equal to X?
- Operation `2 I X`: Update the value of the coefficient A_i to be X.

Can you help Charles with his experiments?

Input

The first line of the input contains the integer N ($1 \leq N \leq 10^5$), the number of positions in the quantum field. The second line contains N positive integers $A_1, A_2, \ldots, A_N$ ($1 \leq A_i \leq 10^5$).

The third line of the input contains the integer Q ($1 \leq Q \leq 10^5$), the number of queries. The next Q lines will contain operations. Each line is an operation identified by the integer T ($1 \leq T \leq 2$). Operations with $T = 1$ are followed by an integer X, and operations of type 2 are followed by the integers I ($1 \leq I \leq N$) and X. In both operations, $1 \leq X \leq 10^5$.

Output

For each experiment conducted by Charles, print the probability $\frac{P}{Q}$ that the experiment has beauty equal to X in the form $P \cdot Q^{-1} \pmod{998244353}$. It is guaranteed that P is a non-negative integer and Q is a positive integer, and that Q^{-1} with the property $QQ^{-1} = 1 \pmod{998244353}$ exists.

Sample input 1	**Sample output 1**
4 1 2 4 8 6 1 1 1 2 1 4 1 8 1 3 1 5	931694730 465847365 732045859 865145106 0 0

Explanation of sample 1:

The 15 possible paths that can be taken in an experiment are:

- $\{a_1\} = \{1\} \to \gcd(1) = 1$
- $\{a_2\} = \{2\} \to \gcd(2) = 2$
- $\{a_3\} = \{4\} \to \gcd(4) = 4$
- $\{a_4\} = \{8\} \to \gcd(8) = 8$
- $\{a_1, a_2\} = \{1, 2\} \to \gcd(1, 2) = 1$
- $\{a_1, a_3\} = \{1, 4\} \to \gcd(1, 4) = 1$
- $\{a_1, a_4\} = \{1, 8\} \to \gcd(1, 8) = 1$
- $\{a_2, a_3\} = \{2, 4\} \to \gcd(2, 4) = 2$
- $\{a_2, a_4\} = \{2, 8\} \to \gcd(2, 8) = 2$
- $\{a_3, a_4\} = \{4, 8\} \to \gcd(4, 8) = 4$
- $\{a_1, a_2, a_3\} = \{1, 2, 4\} \to \gcd(1, 2, 4) = 1$
- $\{a_1, a_2, a_4\} = \{1, 2, 8\} \to \gcd(1, 2, 8) = 1$
- $\{a_1, a_3, a_4\} = \{1, 4, 8\} \to \gcd(1, 4, 8) = 1$
- $\{a_2, a_3, a_4\} = \{2, 4, 8\} \to \gcd(2, 4, 8) = 2$
- $\{a_1, a_2, a_3, a_4\} = \{1, 2, 4, 8\} \to \gcd(1, 2, 4, 8) = 1$

where $\gcd(X)$ is the greatest common divisor of the set X.
Thus:

- The probability of the beauty of an experiment being equal to 1 is $\frac{8}{15}$.
- The probability of the beauty of an experiment being equal to 2 is $\frac{4}{15}$.
- The probability of the beauty of an experiment being equal to 4 is $\frac{2}{15}$.
- The probability of the beauty of an experiment being equal to 8 is $\frac{1}{15}$.
- The probability of the beauty of an experiment being equal to 3 is $\frac{0}{15}$.
- The probability of the beauty of an experiment being equal to 5 is $\frac{0}{15}$.

Sample input 2	**Sample output 2**
3 18 29 15 5 1 12 1 25 1 28 2 1 25 1 5	0 0 0 855638017

Explanation of sample 2:

After the operation `2 1 25`, the sequence of coefficients becomes $(25, 29, 15)$. Before the type 2 operation, the sequence of coefficients was $(18, 29, 15)$.