ECN Programming Contest, November 25, 2023

A — Gifts

Fanurie wants to visit each of his N friends and leave a gift of a certain value. He drew a map in the form of an undirected tree with N vertices and found a rule by which he would visit them.

From a node X, Fanurie does the following operations:

- leave the gift ready for node X;
- go through unvisited nodes in ascending order;
- returns to the node from which he arrived in X.

Fanurie decides to always start from node 1.

In order not to upset his friends, he decided to compensate for the delay by increasing the value of the gift. Thus, for each $2 \le i \le N$, $v[i]$ is known – by how much the value of the gift received by the ith visited friend increases compared to the value of the gift received by the i – 1-th visited friend. It is considered that the first friend visited receives the value $v[1]$.

Fanurie defines the subtree induced by a list of nodes as the set of nodes in the minimal subtree that contains each node in the list. It also defines the value of a list of nodes as the sum of the values of the gifts left in the nodes.

Fanurie is asking you to help him answer Q queries of the form $S\ K\ a_1\ a_2\ \ldots\ a_K$ with the following meaning:

Which is the node in the subtree induced by the K elements that added to the list of nodes would make the value of the list as close as possible to S? You can not add a node that is already present in the list. If there are several options, choose the smallest node.

Fanurie promises to give you a priceless gift if you help him.

Input Specification

The first line contains two integers N and Q. The second line has N values, the elements of v. Each of the following N – 1 lines have two integers, meaning that there is an edge between them. The following Q lines describe the questions in the form $S\ K\ a_1\ a_2\ \ldots\ a_K$.

Input Limits and Constraints

- $1 \le N \le 100\,000$;
- $1 \le Q \le 50\,000$;
- $1 \le v[i] \le 1\,000$, $1 \le i \le N$;
- $2 \le K \le 3$;
- $1 \le S \le 1\,000\,000\,000$;
- it is guaranteed that there are at least $K + 1$ nodes in any induced subtree.

Output Specification

Print one line to the standard output with Q space-separated integers, representing the answer to each query. Do not put a space after the last number!

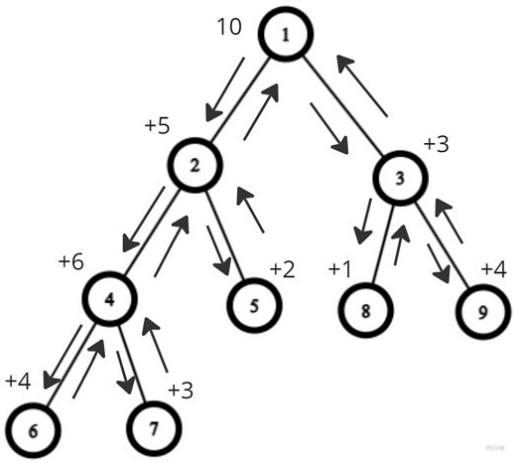
Sample Input

```
9 3
10 5 6 4 3 2 3 1 4
1 2
1 3
2 4
2 5
4 6
4 7
3 8
3 9
73 2 5 6
34 1 3
1000000000 2 7 8
```

Output for Sample Input

```
2 8 9
```

Explanation



In the first query the induced subtree has the following node-value pairs: {(2,15), (4,21), (5,30), (6,25), (7,28)}. Both nodes 2 and 4 lead to a list value at a distance 3 from *S* (which is minimum). In the second example node 8 leads to a list value equal to *S*.

ECN Programming Contest, November 25, 2023

B — Colors in Store

A new shipment of n tablecloths has arrived at a store. Each tablecloth is described by three attributes: price (p_i), front color (a_i), and back color (b_i), where p_i is an integer and represents the cost of the ith tablecloth, and a_i and b_i are integers ranging from 1 to 3, indicating the colors of the front and back of the tablecloth, respectively. It is guaranteed that all p_i values are unique.

The store expects m customers to visit, and each customer intends to purchase exactly one tablecloth. For each customer, we know their favorite color (c_j). A customer will buy a tablecloth if either the front or back side matches their favorite color. In such cases, the customer selects the cheapest available option among the tablecloths that match their preferred color. If there are no tablecloths with an acceptable color for the customer, they won't make a purchase. The customers arrive sequentially, and each customer is served only after the previous one has been attended to.

Your task is to calculate the price each customer will pay for the tablecloth they choose.

Input Specification

The first line contains an integer n ($1 \le n \le 200\,000$), representing the number of tablecloths. The following line provides a sequence of n integers: $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 1\,000\,000\,000$), indicating the prices of the tablecloths. The next line contains a sequence of n integers: $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 3$), representing the front colors of the tablecloths. The subsequent line contains a sequence of n integers: $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 3$), indicating the back colors of the tablecloths. Then, there's a single integer m ($1 \le m \le 200\,000$), denoting the number of customers. The next line provides a sequence of m integers: $c_1, c_2, \ldots, c_m$ ($1 \le c_j \le 3$), signifying the favorite color of each customer. The customers arrive in the order they are listed in the input, and each customer is served only after the previous one has been assisted.

Output Specification

Print one line to the standard output with m space-separated integers, where the jth integer should represent the price of the tablecloth that the jth customer will purchase. If the jth customer decides not to buy anything, print –1 for that customer.

Sample Input

```
5
300 200 400 500 911
1 2 1 2 3
2 1 3 2 1
6
2 3 1 2 1 1
```

Output for Sample Input

```
200 400 300 500 911 -1
```

ECN Programming Contest, November 25, 2023

C — Golden Primes

A prime number is called a golden prime when it takes the following form:

$$p = \Phi^2 - \Phi - 1,$$

where $\Phi = 2^n$, and n is a positive integer.

Your task is to find golden primes. Write a program that calculates all values for n for which p is a golden prime less than a given value b.

Input Specification

The input consists of a single positive integer $b < 2^{1000}$.

Output Specification

Your program must print to the standard output, on separate lines, all n values found.

Sample Input

```
1024
```

Output for Sample Input

```
2
4
5
```

ECN Programming Contest, November 25, 2023

D — Egyptian Fractions

Mathematics is ancient. Very ancient. Imagine yourself more than 3500 years ago in ancient Egypt. An Egyptian fraction is a finite sum of distinct unit fractions, like this:

$$\frac{1}{5}+\frac{1}{24}+\frac{1}{840}.$$

The sum of these fractions is $^{17}/_{70}$. It may not be immediately evident that every fraction can be represented as a sum of distinct unit fractions, making it, too, an Egyptian fraction. However, these decompositions are not necessarily unique. For instance,

$$\frac{17}{70}=\frac{1}{5}+\frac{1}{24}+\frac{1}{840}=\frac{1}{7}+\frac{1}{10}.$$

The practicality of Egyptian fractions becomes apparent when you imagine the scenario of paying $^{17}/_{70}$ of a quantity. Instead of repeating the fraction $^{1}/_{70}$ 17 times, the Egyptians devised a more efficient approach. By preparing in advance some unit fractions, they could simply present two of them: $^{1}/_{7}$ and $^{1}/_{10}$. This demonstrates how the use of Egyptian fractions offered a more practical solution in such situations. While the use of Egyptian fractions proved practical in certain scenarios, a drawback arises from the challenge of decomposing a fraction into a sum of distinct unit fractions. Many related questions remain open today. For instance, there is a conjecture, not yet proven, suggesting that for every $n \geq 5$, the fraction $4/n$ can be decomposed into a sum of not more than 3 unit fractions (Erdős–Straus conjecture, verified up to $n < 10^{17}$).

A natural question arises: given a fraction, what is the minimum number of unit fractions in its Egyptian number representation? Currently, it remains unknown whether a polynomial time algorithm exists for this problem, and the computational complexity is also uncertain. While we can resort to a brute-force search for computation, it's an open challenge to devise a more efficient algorithm.

Your task now is to design an algorithm capable of computing the Egyptian fraction representation(s) of a given fraction with the fewest possible terms. Assume the fraction is composed of positive integers and has a value less than 1. Your goal is to compute all its expansions as a sum of unit fractions with the minimum number of terms.

Input Specification

The input contains several lines, each has a fraction written in the form a/b, where a and b are positive integers, $0 < a < b$.

Output Specification

The output should have the same number of lines as the input. For each input line, print one line to the output, containing a number, the minimum number of unit fractions, followed by the space-delimited denominators of the minimum-length expansion enclosed in parentheses. If the expansion is not unique, additional denominators should be listed after a comma and a space. Ensure that the denominators and parentheses are ordered in increasing lexicographical order for each line.

Sample Input

```
33/182
223/12342
511/1404
```

Output for Sample Input

```
2 (7 26)
2 (102 121)
3 (3 36 351), (4 9 351), (4 13 27)
```

ECN Programming Contest, November 25, 2023

E — Lanovka – The Cable Car

There are N peaks in the Vector Vista Mountains. We want to install cable car columns on K adjacent peaks so that if the largest height of the cable car (peak + column) is H, then on the other peaks the heights should be $H - 1$, $H - 2$, …, $H - K + 1$, so that the cable car continues to go up smoothly. The cable car always starts from the right and goes to the left.

We can achieve this by increasing the height of the cable car columns by 1 unit. By how many minimum units must the heights of columns be increased in total in order to create a cable car of length K?

Input Specification

The first line of the input contains the number of test cases T ($1 \le T \le 20$).

This is followed by two lines for each test case. The first line contains the number of peaks N ($1 \le N \le 200\,000$) and the length of the cable car K ($1 \le K \le N$). The second line contains N natural numbers H_i ($1 \le i \le N$), the heights of the mountain peaks ($1 \le H_i \le 1\,000\,000\,000$).

Output Specification

For each test case, print a single number on a line by itself: the minimum number of units.

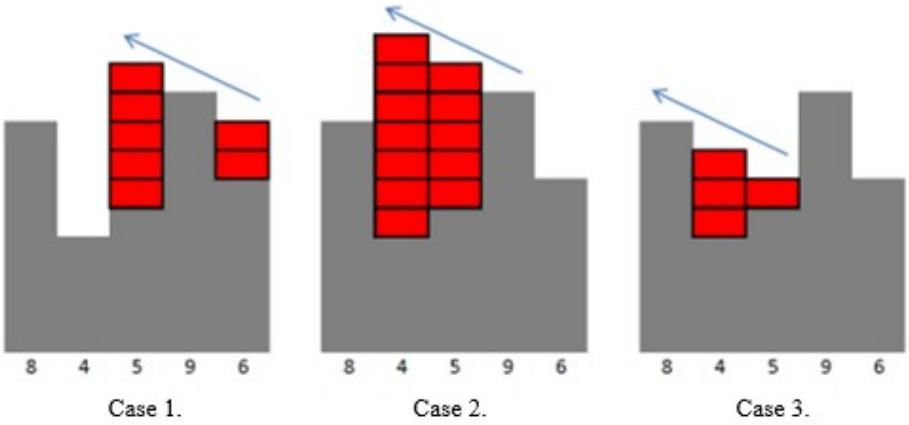
Sample Input

```
1
5 3
8 4 5 9 6
```

Output for Sample Input

```
4
```

Explanation



The 3 possible cable cars are:

- Case 1: Peaks 5, 9, 6, with heights 10, 9, 8, and differences 5, 0, 2. The sum is 7.
- Case 2: Peaks 4, 5, 9, with heights 11, 10, 9, and differences 7, 5, 0. The sum is 12.
- Case 3: Peaks 8, 4, 5, with heights 8, 7, 6, and differences 0, 3, 1. The sum is 4.

The minimum of the total differences is 4.

ECN Programming Contest, November 25, 2023

F — Kings Again

Last year's problem with kings was too easy, so I made it more difficult. Determine the number of ways in which k kings can be placed on an $n \times m$ chessboard without attacking each other. Two kings attack each other if they are on adjacent cells horizontally, vertically, or diagonally.

Input Specification

The first line of the input contains t, the number of test cases to follow ($1 \le t \le 15\,000$). Each test case is described by the values of n, m, and k, in this order, separated by spaces ($1 \le n \le m \le 15$, $1 \le k \le 100$).

Output Specification

For each test case, you must output a single line, the number of ways modulo $10^9 + 7$.

Sample Input

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

Output for Sample Input

```
3
1
79
```

ECN Programming Contest, November 25, 2023

G — Breaking a Quantum Cryptography Machine

During the Third Interstellar War, the Secret Service tasked you with cracking the enemy's quantum cryptography machine that protects the enemy's communications. The last captured cryptographic messages consist of instructions intercepted by the Secret Service and their corresponding solutions. The correctness of the messages have been verified by the Secret Service. Find out the operating principle of the quantum machine using the messages broken so far, and reimplement it using C/C++.

Captured Instructions #1

```
1. TITKOS_P 3
2. TITKOS_P 13
3. TITKOS_A
4. TITKOS_P 5
5. TITKOS_S
6. TITKOS_E
```

Solution #1

```
1. -11
```

Captured Instructions #2

```
1. TITKOS_P 2
2. TITKOS_P 8
3. TITKOS_P 23
4. TITKOS_S
5. TITKOS_A
6. TITKOS_E
```

Solution #2

```
1. 17
```

Captured Instructions #3

```
1. TITKOS_P 2
2. TITKOS_P 8
3. TITKOS_P 23
4. TITKOS_A
5. TITKOS_S
6. TITKOS_E
```

Solution #3

```
1. 29
```

Input Specification

The input contains a sequence of similar instructions to those in the captured messages, one instruction per line. The input may be considered correct in all cases. There are no conflicting instructions or instructions that cannot be executed at any point in time. All intermediary numbers as well as the results are integers that fit into the range of the `int` type.

Output Specification

Print one line to the standard output, containing the solution of the sequence of instructions read from the input.

ECN Programming Contest, November 25, 2023

H — “Optimizing” Ascent: Navigating Bear-Dangerous Territory in a Binary Matrix Mountain

Let's imagine a “mountain” with n levels (1, 2, …, n), where the levels (or “mountain slices”) are encoded by $m \times m$ binary matrices. In each matrix, the value 1 indicates points belonging to the mountain. Both 1s and 0s form a connected region, considering two elements adjacent if they share a common side. Other two constraints are: (i) the mountain slices are “concentric” in the sense that in the positions of 1s at level i (i = 2, …, n), there are also 1s at level (i – 1); (ii) the levels have no common boundary points. In the last-level matrix, there is a single 1 representing the peak. Since all internal points at each level are considered bear-dangerous, we want to climb to the summit using the following method: (i) we can start from any boundary/edge point of level 1; (ii) from the edge of each level to the edge of the next level, we want to reach in the minimum number of steps, ultimately reaching the summit at the top level; (iii) climbing to the next level or moving along the edge of any level is considered safe (these areas are avoided by bears, as they have a fear of heights). If we follow the strategy outlined above, how many steps do we take in dangerous territory from the base to the summit?

Input Specification

The first line of the input contains t, the number of test cases. For each test case: (i) the first line contains n and m, the number of levels and the size of the square matrices ($1 < n < 100$, $3 < m < 100$); (ii) each of the next $n \cdot m$ lines contains m binary values (0/1), separated by one space, giving the elements of the matrices.

Output Specification

For each test case, print a single line to the standard output, containing the number of steps taken in dangerous territory.

Sample Input

```
1
3 10
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 0 1 1 1 1 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Output for Sample Input

```
2
```

Explanation

The sample input consists of a single test case (t = 1) corresponding to a 3-level mountain coded by 10 × 10 matrices. Bold 1s represent edge points. Underlined 1s represent the accessed dangerous points. At level 1, the edge points of level 2 are accessed by 1 internal step (step (3,9)–(3,8)), and at level 2, the position of the peak point from level 3 is also accessed by 1 internal step (step (5,5)–(5,6)).

ECN Programming Contest, November 25, 2023

I — Hamilton

Santa Claus has decided this year to start delivering Christmas gifts on time, with the help of his elves.

His favorite elf, named Hamilton, needs to traverse a 10000-meter-long, 15-meter-wide stretch of road, where the coordinates (x, y) of each mailbox are precisely known ($0 \le x \le 10\,000$, $0 \le y \le 15$, x and y are natural numbers).

For any two mailboxes (x_a, y_a) and (x_b, y_b), the abscissas are different ($x_a \neq x_b$). And, of course, this condition cannot be fulfilled for the ordinates y_a and y_b. Hamilton, the elf, has to distribute approximately 4000 packages and would like to carry the packages to mailboxes in such a way that he travels the **shortest distance in a single round trip**, placing each package in the corresponding mailbox. In other words, the abscissas of the mailboxes continuously increase in the first segment and then continuously decrease. The length of the road should be measured from the left side, starting from the first mailbox, and it should end in the start point.

The requirement for the task is to calculate, given the coordinates of the mailboxes, the shortest distance starting from the first mailbox.

Input Specification

The first line of the input contains the number of mailboxes n. This is followed by n lines, each containing two natural numbers x and y, separated by a space, representing the coordinates of a mailbox.

Output Specification

A single real number should be printed, representing the length of the shortest route, with exactly 6 decimals.

Sample Input

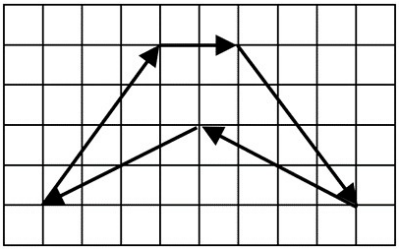
```
5
4 5
1 1
6 5
5 3
9 1
```

Output for Sample Input

```
20.944272
```

Explanation

Hamilton, the elf, travels the following path from mailbox to mailbox:



5 + 2 + 5 + 4.47213595 + 4.47213595 = 20.944272 (with 6 decimals precision).

ECN Programming Contest, November 25, 2023

J — Find the Identical Twins and Triplets

The real story (three identical strangers)

Starting with 1960, the psychiatrist Peter Neubauer conducted a controversial study about identical twins and triplets adopted apart. The purpose of this study was to determine which factor is more important in development of a human being: genetics or environmental. Twenty years after, Robert Shafran, David Kellman, and Eddy Galland found one another through sheer coincidence and discovered they were separated after their birth. After that, Peter Neubauer ceased the study, all the collected data have been sealed at Yale University and can't be opened until 2066.

The problem to be solved

Given a database of population, your task is to identify every group of identical twins or triplets (sharing the same DNA) and display the members of every group where at least one member is adopted.

Input Specification

The database of population includes some pieces of information about each person, from which we selected only the necessary ones. The database is described as follows:

- The first line contains the number of persons n ($n \le 160\,000$).
- Each of the next n lines contains the following information about each person:
  - personal code (31-bit positive integer)
  - DNA signature (≤ 11 uppercase letters)
  - the character "A" if the person is adopted, or "-" (minus) otherwise
  - name of the person (≤ 27 characters including whitespace)

After inserting a group of twins or triplets, not used positions between consecutive personal codes (considering the ascending order) will never be assigned. For example, in the case of the sample input (see below), 128, 130, and 131 will never be assigned.

Output Specification

After identifying all groups of identical twins or triplets, display the members (with personal code and name separated by a space) of each group where at least one member is adopted. Insert an empty line between any two consecutive groups. The list should be ordered by personal code.

Sample Input

```
10
132 TCA - Lim Leo
135 GAT - Gal Vera
101 ACG A Ban Remo
105 ACT - Kan Mia
107 AGC - Mir Ando
111 AGC A Nor Ken
127 TCA - Lim Nico
103 ACT - Kor Ema
129 TCA A Pat Sam
124 TAC - Gor Alex
```

Output for Sample Input

```
107 Mir Ando
111 Nor Ken

127 Lim Nico
129 Pat Sam
132 Lim Leo
```

Explanation

The following groups are identified:

- 103, 105: no one is adopted, so we don't display this group.
- 107, 111: at least one member is adopted.
- 127, 129, 132: at least one member is adopted.

ECN Programming Contest, November 25, 2023

K — Dependencies

Amazon Web Services (AWS) CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want, and CloudFormation takes care of provisioning and configuring those resources for you. In a CloudFormation template, you specify the logical names of resources, together with their properties, including the dependent resources. The logical name (or logical ID) must be alphanumeric and unique within the template. The logical name is used to reference to the resource in other parts of the template.

In the following CloudFormation template, such logical names are `MyEC2Instance` and `MyEIP`:

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template
Resources:
  MyEC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
  MyEIP:
    Type: 'AWS::EC2::EIP'
    Properties:
      InstanceId: !Ref MyEC2Instance
```

Write a program that reads logical names of resources given in an AWS CloudFormation template from the standard input. For each logical name, you will be given all the other names of resources that are required to exist for the resource to be created. Your program has to determine whether a given sequence of resources can or cannot be created in the specified order, and if not, the program also has to give the logical names of all resources in the sequence that cannot be created.

Input Specification

The first line of the input contains two integers R and S, representing the number of resources and the number of sequences, respectively ($1 \le R \le 100$, $1 \le S \le 100$). Each of the following R lines describes a resource and its dependencies (in no particular order). Each resource is represented by an identifier, which is a string of at most 20 alphanumeric characters. The first resource in each line is followed by a colon (“:”), and then optionally a space-separated list of its dependencies, also being resource identifiers. Finally, the last S lines of the input contain the sequences, in the form of space-separated resource IDs. Both the resource descriptions and the sequences will contain only valid resource IDs.

Output Specification

For each sequence in the input, print one line to the output.

If there are at least two resources in the sequence that cannot be created in the given order ("problematic" resources), then print "`Problematic resources: `", followed by a space-separated list of the logical names of all the problematic resources in the same order as they appeared in the sequence.

If there is only one problematic resource, print the message "`Problematic resource: `", followed by the logical name of the problematic resource.

If all the resources can be created in the order presented in the sequence, then print "`No problematic resources.`".

Sample Input 1

```
2 2
MyEC2Instance:
MyEIP: MyEC2Instance
MyEC2Instance MyEIP
MyEIP MyEC2Instance
```

Output for Sample Input 1

```
No problematic resources.
Problematic resource: MyEIP
```

Sample Input 2

```
5 2
A: B C D
B: D
C:
D:
E: A B
A B C D E
D B C A E
```

Output for Sample Input 2

```
Problematic resources: A B E
No problematic resources.
```

ECN Programming Contest, November 25, 2023

L — Windmill Lottery

In *Windmillia*, there is a special national lottery system. Everyone can get a number and participate in countless draws. The numbers owned by the players are put in two lines (*Line*1 and *Line*2) based on the number specification, and they will belong to that line forever. The numbers and draws are generated based on the following rules:

- $LineHelper_n = (((La * LineHelper_{n-1} + Lb) * Lc) + Ld) \ mod \ (10^9 * 7)$
- $Line_n = LineHelper_n \ mod \ 3 + 1$

  The $Line_n$ number can be 1 or 2, which is used to insert a number into the specific line, or 3, which means performing draw if it is possible.

- $Number_n = (((a * Number_{n-1} + b) * c) \ \% \ d)$

  If $Line_n$ is 1 or 2, then $Number_n$ is used as a number. Else, $Number_n$ is used as *rotationCount* for the draw.

- $Offset_i = (((La * Offset_{i-1} + Lb) * Lc) + Ld) \ mod \ 21 - 10$

  Offset is computed only when $Line_n$ is 3.

A draw can take place at any time if the count of the numbers in the two lines have the same parity. A draw is performed as follows:

- The two lines are being sorted.
- The middle point of *Line*1 and *Line*2 is placed on each other (in case of even count number, a virtual center is selected in both lines, between the two middle numbers).
- A rotation count is set (*rotationCount*) and *Line*2 is rotated around the middle point *rotationCount* times by 180 degrees, while *Line*1 stays fixed.
- From now on, we consider the middle point being the middle point of the two lines if the number of elements in either is an odd number; otherwise, middle point is the middle-left point if $Offset_i < 0$, and middle-right point if $Offset_i \geq 0$, in both lines.
- We take the two numbers with $Offset_i$ offset from the middle points, and calculate their sum modulo m, where m is the maximum value found already in the two lines. If a line is too short to have a corresponding position, we can take 0 for that number.
- The computed value will be the winning number of the draw.

Input Specification

The first line contains the values of a, b, c, d, and the starting value for *Number* (five space-separated numbers). The second line contains the values of La, Lb, Lc, Ld, and the starting value for *LineHelper* (five space-separated numbers). The third line contains the initial value of *Offset* and the value of n, the overall number of numbers generated, separated by a space.

Input Limits and Constraints

- $1 \le$ *lottery numbers* $\le 10^{10}$;
- $0 \le$ *lottery numbers count* $\le 10^{6}$;
- $-10 \le$ *Offset* ≤ 10;
- $0 \le$ *rotationCount* $\le 10^{10}$;
- $1 \le a, b, c, d, La, Lb, Lc, Ld \le 10^{5}$.

Output Specification

The output should contain the result of every draw, one line for every number.

Sample Input

```
1 1 1 101 123
1 1 1 1 1
2 30
```

Output for Sample Input

```
54
```

Explanation

After generating 30 numbers, the two arrays are sorted, because a draw is generated. After sorting, they are placed so that the middle points are in the same column:

*Line*1: 23 26 29 32 35 38 41 **44** 47 50 53 56 59 62 65

*Line*2: 25 28 31 34 37 40 43 **46** 49 52 55 58 61 64 67

The draw is generated with *Offset* = –6 and *rotationCount* = 68. *Line*2 is rotated 68 times, and the –6th elements relative to the middle points are added together: (26 + 28) = 54 (*mod* 67). The result is printed. There was only one draw.

ECN Programming Contest, November 25, 2023

M — Modputer

Modputer is a strange mathematical computing machine that works as follows. The machine has an internal state, which is represented by an array A of N positive integers. The input of the machine is a list of M positive integers, each greater than 1.

The machine reads the integers on the input one by one. For each integer D in the input, the state of the machine gets updated: **every** integer in the state array which is divisible by D increases by one.

It is difficult to simulate the transitions of Modputer using a regular computer. Your task is to count the number of times each integer in the state array got updated while processing the input of the machine and report the total number of changes.

More formally, let $A[i]$ denote the value of the ith element of the state array before reading the input, and $B[i]$ denote the value of the same element after processing the input. You must calculate the sum of $B[i] - A[i]$ over all i.

Input Specification

The first line of the input contains N and M, the number of elements in the state array and the length of the input list ($1 \le N, M \le 100\,000$).

Each of the next N lines contains an integer $A[i]$, the initial values present in the state array ($1 \le A[i] \le 300\,000$). Each of the next M lines contains an integer $D[i]$, the input values ($2 \le D[i] \le 300\,000$).

Output Specification

Print one line to the standard output, containing the total number of changes in the values of the state array while processing the input.

Sample Input

```
3 5
10
11
12
2
11
4
13
2
```

Output for Sample Input

```
1. 12
```

Explanation

The initial state array is [10, 11, 12].

After reading the value D[0] = 2, the state changes to [11, 11, 13].

After reading the value D[1] = 11, the state changes to [12, 12, 13].

After reading the value D[2] = 4, the state changes to [13, 13, 13].

After reading the value D[3] = 13, the state changes to [14, 14, 14].

After reading the value D[4] = 2, the state changes to [15, 15, 15].

The first state value was updated 5 times, the second 4 times and the third 3 times. The total number of changes is 5 + 4 + 3 = 12.

ECN Programming Contest, November 25, 2023

N — Carpathian Riders

Carpathian Riders is a famous motorcycle gang and competitive programming club. They decided to participate in the upcoming ECN Sapientia programming contest, travelling there on their bikes. To get to the contest venue, they must cross the Carpathian Mountains.

The Carpathians is a grid of R rows and C columns. Each cell has an elevation value between 0 and 1000, inclusive. Some cells are not passable on motorbikes – those cells are marked with the elevation value –1.

The gang can enter the Carpathians at any passable cell on the western edge of the grid and leave the mountains at some cell on the eastern edge. They travel from the west to the east in multiple steps. In each step, they can move to the cell either directly to the east, or diagonally to the northeast, or diagonally to the southeast. Each cell that they visit must be passable.

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

A sample grid representing the Carpathian Mountains.

The gang does not want to be late from the contest, so they try to avoid riding up to cells with high elevation: the sum of elevations during the trip must be minimized.

Sounds easy enough? Well, there's a catch. Since this is a motorcycle gang, they love visiting *mountain passes*. A mountain pass is a grid cell that has a strictly greater elevation value than the two cells to its east and west, but has a strictly lower elevation value than the two cells to its north and south. Cells on the edges of the grid cannot be mountain passes, similarly to cells adjacent to impassable cells. In the example above, all cells that are mountain passes are shaded in grey.

The bikers decided that they want to visit **exactly** P passes along the trip. Your task is to help them planning a trip from the west to the east which contains exactly P passes, avoids impassable cells and the sum of elevations is minimized.

Input Specification

The first line of the input contains the number of rows and columns R and C ($3 \le R, C \le 500$) and the exact number of passes to visit P ($0 \le P \le 10$).

Each of the next R lines contains C elevation values. Impassable locations are represented by –1, and all other elevations are between 0 and 1000, inclusive. There is at least one cell on the western and one cell on the eastern edge of the grid that are passable.

Output Specification

Print one line to the standard output, containing the sum of elevations along an optimal trip with exactly *P* passes. If no such trip exists, output the string "`impossible`".

Sample Input 1

```
5 7 2
-1 -1 2 5 4 3 1
3 4 1 4 1 2 1
3 4 5 5 3 4 5
2 3 2 1 2 3 2
-1 5 4 1 4 4 2
```

Output for Sample Input 1

```
14
```

Sample Input 2

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

Output for Sample Input 2

```
impossible
```

Explanation

The first example corresponds to the example in the statement. An optimal trip consists of the cells with elevations 2 – 3 – 2 – 1 – 3 – 2 – 1.

In the second example, there are no mountain passes in the grid, so it is not possible to plan a trip that contains exactly 1 pass.