培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE

第九屆 · 2025

2025 年 8 月 13 日（星期三）

香港培正中學

名稱			時間限制	記憶體限制
A		Advanced Preparations	0.5 秒	256 MB
B		Broken Joy-Con	0.5 秒	256 MB
C		Copper Mining	0.5 秒	256 MB
D		Diving Groupmates	1 秒	256 MB
E		Express Trains	0.5 秒	256 MB
F		Faster Route	0.5 秒	256 MB
G		Greedy Merchants	1 秒	256 MB
H		Hurricane Signal	0.5 秒	256 MB
I		Interesting PuiLaTiu	0.5 秒	256 MB
J		Jumping Game	0.5 秒	256 MB
K		Keep Sorting	0.5 秒	256 MB
L		Linesweeper	0.5 秒	256 MB

Pascal	`ppcx64-3.0.0 -O2 -Sg -v0 -dEVAL -XS`
C	`gcc-7 -static -Wno-unused-result -DEVAL -lm -s -O2`
C++	`g++-7 -static -std=c++17 -Wno-unused-result -DEVAL -lm -s -O2`
Python 3	`python3.5 -O -S`

編繹時間限制 10 秒　編繹記憶體限制 512 MB

作者/編製　鄭希哲 鄭羽辛 招朗軒 何翊蓁 郭正謙 盧沂鋒 龍瑞希
黃卓翹 黃進 黃浩恩 黃敏恆 黃梓謙 楊汶璁 楊承羲 袁樂勤

NOTES

1. Output strings are checked case-sensitively unless otherwise specified.
2. Trailing spaces in output lines are allowed.
3. The end-of-line character at the end of the output is optional.
4. Using scientific notation for real number output is not allowed.
5. 64-bit integer types (e.g. `int64` for Pascal, `long long` for C/C++) may be required for some problems. The C format string token is `%lld`.

A Advanced Preparations

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE
第九屆 · 2025年8月13日

The city of Aetherrain is notorious for its unpredictable weather. To protect students from dangerous floods, the mayor has announced:

"If the rainfall of any day is higher than or equal to 200 mm, classes the next day are suspended!"

Importantly, if classes are suspended and that day's rainfall is also ≥ 200 mm, then the next day's classes are suspended too.

When classes were suspended but the day had 0 mm rainfall, it's a **Failed Prediction** — angry parents complain about taking a day off for nothing while still paying school fees.

You, as a Raincaller, want to create as much chaos as possible. Your task is to plan a rainfall schedule that maximizes the number of days of **Failed Predictions** for the next N days with a total rainfall of exactly S mm, and the amount of rainfall on each day must be a **non-negative integer**. To avoid severe flooding, the rainfall on a single day should not exceed 350 mm.

Any schedule that maximizes the number of **Failed Predictions** would be accepted.

Input

The first line contains two integers N $(1 \leq N \leq 100)$ and S $(0 \leq S \leq 350 \cdot N)$, representing the number of days and the total rainfall of the N days respectively.

Output

The output only contains one line, a construction of rainfall for the next N days that attains this maximum value, where the i-th integer represents the amount of rainfall on day i. If there are multiple solutions, output any of them.

Examples

input	output
5 795	307 0 154 334 0

input	output
4 1300	350 350 350 250

Note

In sample 1, the schedule has attained its maximum number of **Failed Predictions**. In the constructed schedule,

- On day 1, the rainfall is 307 (≥ 200) mm. The classes on day 2 are suspended.
- On day 2, the rainfall is 0 mm. The classes are suspended but the rainfall is 0 mm. Thus, a **Failed Prediction** is recorded.
- On day 3, the rainfall is 154 (< 200) mm. Therefore, the classes on day 4 are not suspended.
- On day 4, the rainfall is 334 (≥ 200) mm. The classes on day 5 are suspended.
- On day 5, the rainfall is 0 mm. The classes are suspended but the rainfall is 0 mm. Thus, another **Failed Prediction** is recorded.

In sample 2, it can be shown that it is impossible to construct a plan that contains any **Failed Prediction**. In the constructed schedule,

- On days 1 to 3, the rainfall is 350 mm each day. The amount of rainfall must not exceed 350 mm, as it would cause city flooding.
- On day 4, the rainfall is 250 (≥ 200) mm. However, since day 5 is out of the schedule, no further **Failed Predictions** can occur.

B Broken Joy-Con

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Ben is playing *Mario Kart World* on his brand new Nintendo Switch 2! In a moment of excitement, Ben accidentally dropped his Joy-Con, and now both the joystick and the gyroscope are broken. This means his character can only move forward in a straight line from now on. Ben wants to know if it's still possible to complete certain tracks with this limited control scheme.

You are given a map of a race track, which is represented as a rectangular grid with R rows and C columns. Let's define (X, Y) as the cell located at the X-th row and the Y-th column. Ben starts at the cell $(1, 1)$ and is initially facing east, meaning that he travels towards increasing column indices. The finishing line is located at cell (R, C). Due to his broken controller, Ben can only perform the following move: advance a cell forward in the direction he is currently facing.

The grid contains K pairs of linked portals. When Ben drives onto one portal of a pair, he is instantly transported to the location of the other portal in that same pair. After teleporting, Ben continues moving forward from his new location in the same direction he was already traveling. He does not get teleported back immediately. If Ben's path would take him off the grid, he hits a wall. His kart stops permanently at his last valid cell, and his run is over. He has failed to reach the finish line.

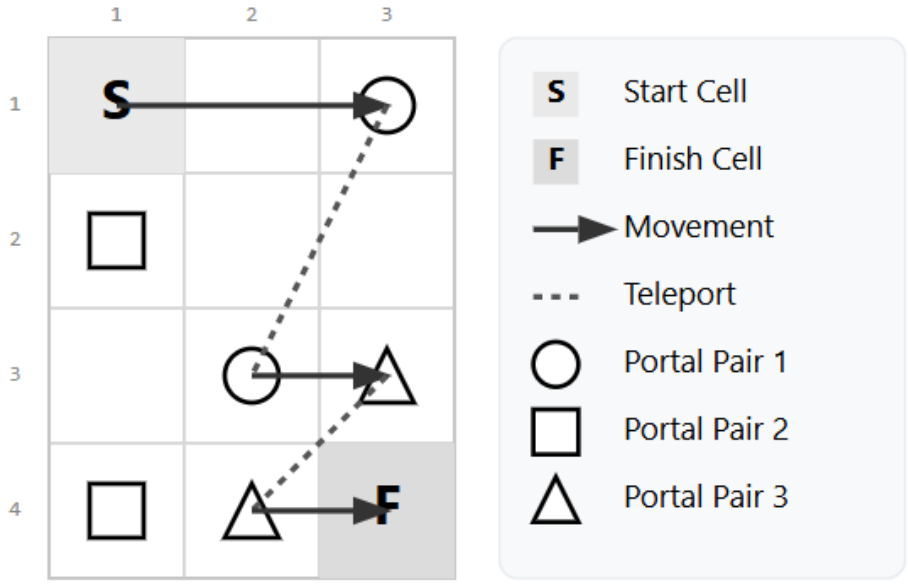


Figure 1

The image above shows a racing track with 3 pairs of teleporter portals. It illustrates how Ben reaches the finishing cell by moving in one direction only.

As Ben's friend, your task is to help him determine if it is possible to reach the finishing line at cell (R, C).

Input

The first line of the input contains two integers: R and C $(1 \leq R, C \leq 10^9)$, the number of rows and columns in the grid, respectively.

The second line contains one integer K $(0 \leq K \leq 3 \times 10^4)$, indicating the number of teleporter portal pairs.

The next $2K$ lines each describe one portal cell. Each line contains three integers: P_r, P_c, P_{id} $(1 \leq P_r \leq R, 1 \leq P_c \leq C, 1 \leq P_{id} \leq K)$, indicating the location of the portal and the pair it belongs to. The portal is located at (P_r, P_c) and belongs to the P_{id}-th pair. It is guaranteed that each P_{id} will exist exactly twice.

Note that the $2K$ portal cells are given in the order where they are first sorted increasingly by row indices, then increasingly by column indices.

Please also note that no two teleporter portals share the same cell and there won't be any portals in the starting cell or in the finishing cell.

Output

Print a single line containing `YES` if it is possible for Ben to reach the finish line, and `NO` otherwise.

You can output the answer in any case (upper or lower). For example, the strings `yEs`, `yes`, `Yes`, and `YES` will be recognized as positive responses.

Examples

input

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

output

```
YES
```

input

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

output

```
NO
```

Note

Please refer to Figure 1 for Sample 1.

In Sample 2, it is impossible for Ben to reach the finishing cell.

C Copper Mining

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Steve is a professional miner who is trying to mine copper ores! Steve is currently in a mine which can be represented as an $N \times N$ grid. Each cell in the grid is identified by a pair (R, C), meaning that the cell is in the R-th row from top to bottom and the C-th column from left to right. Two cells are said to be adjacent if they share a common edge (in the direction of up, down, left, or right). Formally, (R, C) is adjacent to $(R-1, C)$, $(R+1, C)$, $(R, C-1)$, and $(R, C+1)$, if any exist. In the mine, there are **exactly** 3 copper cells, and all the remaining cells are stone cells.

Steve is trying to mine a mine shaft to connect all 3 copper cells by mining some (possibly, zero or all) stone cells **only**. The copper cells are said to be connected if there exists a path of adjacent cells consisting of copper cells and mined cells between any two copper cells.

Please help Steve find a way to connect all 3 copper cells by mining the minimum number of stone cells. If there are multiple ways to achieve this minimum, any of them is accepted.

Input

The first line contains one integer N ($2 \leq N \leq 500$), representing the size of the mine.

The following N lines each contain a string of N characters, describing a row of the mine. The j-th character on the i-th line represents the cell at (i, j). Each character is either `'X'` or `'.'` if the cell at (i, j) is a copper cell or a stone cell, respectively.

It is guaranteed that there are **exactly** 3 copper cells (`'X'`) in the mine.

Output

The output should consist of N lines, each containing a string of N characters, describing the mine after mining the mine shaft. The j-th character on the i-th line represents the cell at (i, j). Each character is either `'X'`, `'.'`, or `'*'` if the cell at (i, j) is a copper cell, a stone cell, or a mined cell, respectively.

After mining, the copper cells must remain at the same locations, but you can change some stone cells to mined cells.

If there are multiple ways to achieve the minimum number of stone cells mined, any of them is accepted.

Examples

input

```
5
...x.
.....
...x.
.x...
.....
```

output

```
...x.
...*.
...x.
.x**.
.....
```

input

```
5
.....
.x.x.
.....
..x..
.....
```

output

```
.....
.x*x.
..*..
..x..
.....
```

input

```
2
xx
x.
```

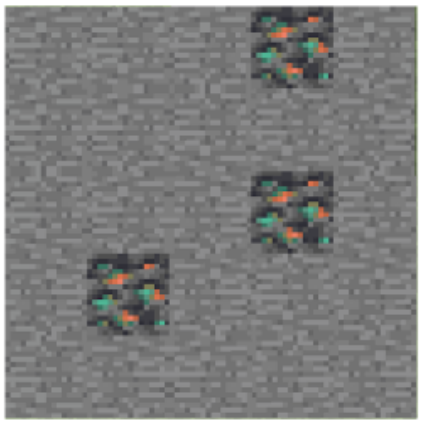
output

```
xx
x.
```

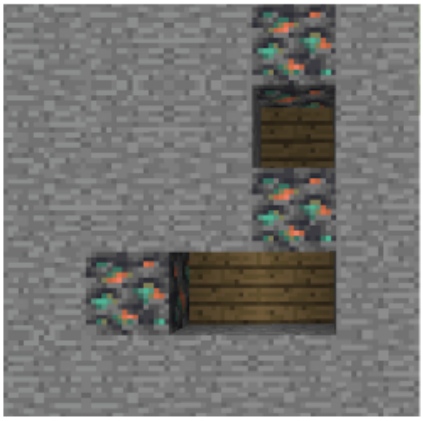
Note

For the first sample test, the copper cells are located at $(1, 4)$, $(3, 4)$, and $(4, 2)$, and the stone cells at $(2, 4)$, $(4, 3)$ and $(4, 4)$ are mined. It can be proven that the minimum number of stone cells needed to be mined to connect the copper cells is 3.

The following diagram shows the mine before and after mining.



The mine before mining



The mine after mining

Note that there are other answers, e.g.

```
...x.
...*.
.**x.
.x...
.....
```

Image source: Minecraft

D Diving Groupmates

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

Ben and his N groupmates, numbered from 1 to N, are collaborating on a heavy school project that consists of M tasks, numbered from 1 to M. To complete the project efficiently and meet the project deadline, Ben decides to assign the M tasks to his N groupmates. For each task i ($1 \leq i \leq M$), there is exactly one groupmate who will be the Person-in-Charge PIC_i ($1 \leq PIC_i \leq N$), responsible for ensuring the task gets done. Additionally, each task i has P_i "peanuters" ("花生友" in Cantonese) — groupmates who prefer to observe the progress and give comments without actively contributing.

To ensure the tasks are completed on time, Ben has to send reminder messages in the group chat, one for each task. The behavior when Ben sends a reminder for task i is as follows:

- The Person-in-Charge PIC_i **dives** — pretends to be unavailable and ignores the reminder message for task i.
- All peanuters of task i **upwater** — pop online and read all messages currently in the group chat. Note that a groupmate can be both Person-in-Charge of some tasks and peanuters of some other tasks. In this case, once they read the message, they will be forced to handle all tasks for which they are the Person-in-Charge among the tasks mentioned in those messages.

Ben's goal is to maximize the number of tasks that are handled by their respective Person-in-Charge. A task is considered handled if the Person-in-Charge reads the reminder message for that task. To achieve this, Ben can carefully decide the order in which he sends reminders for the M tasks. Your task is to help Ben determine the optimal order for sending reminders so that the number of tasks handled by their respective Person-in-Charge is maximized.

Input

The first line contains two integers, N and M, the number of groupmates and the number of tasks respectively. ($1 \leq N \leq 10^5$, $1 \leq M \leq 10^5$)

The second line contains M integers, the i-th integer denotes PIC_i, the Person-in-Charge of task i. ($1 \leq PIC_i \leq N$)

The next M lines each start with an integer P_i specifying the number of peanuters of task i, followed by P_i distinct integers denoting the peanuters of task i. ($0 \leq P_i \leq N-1$, $0 \leq \sum_{i=1}^{m} P_i \leq 10^5$)

It is guaranteed that the Person-in-Charge will not be a peanuter of the same task.

Output

The output contains a line of M integers, representing the optimal order to send the reminder messages that maximizes the number of tasks that can be handled. Each task from 1 to M should appear exactly once.

Examples

input

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

output

```
2 3 1
```

input

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

output

```
4 1 2 3 5
```

Note

Explanation of sample 1:

Consider a project with $N = 3$ groupmates and $M = 3$ tasks.

- Task 1: The Person-in-Charge is groupmate 1 ($PIC_1 = 1$), and the peanuter is groupmate 2.
- Task 2: The Person-in-Charge is groupmate 3 ($PIC_2 = 3$), with no peanuters.
- Task 3: The Person-in-Charge is groupmate 2 ($PIC_3 = 2$), and the peanuter is groupmate 3.

Ben sends reminder messages in the order $[2, 3, 1]$.

1. Ben sends a reminder for task 2.
2. Ben sends a reminder for task 3. The peanuter, groupmate 3, reads all messages in the group chat (reminders for tasks 2 and 3), and handles task 2 since groupmate 3 is the Person-in-Charge for task 2 ($PIC_2 = 3$).
3. Ben sends a reminder for task 1. The peanuter, groupmate 2, reads all messages in the group chat (reminders for tasks 2, 3, and 1), and handles task 3 since groupmate 2 is the Person-in-Charge for task 3 ($PIC_3 = 2$).

Thus, 2 tasks are handled, which is the maximum possible.

E Express Trains

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Bob, a railway enthusiast, is excited about the city's new train line. However, he finds the planned service too slow and has devised his own proposal to improve travel times.

In Bob's proposal, the train line consists of N stations numbered sequentially from 1 to N. Every train station will have an integer level A_i, where $1 \leq A_i \leq M$. There will also be M train types numbered from 1 to M. A train of type j stops only at stations i where the station's level $A_i \leq j$. All trains travel only in the direction of increasing station numbers.

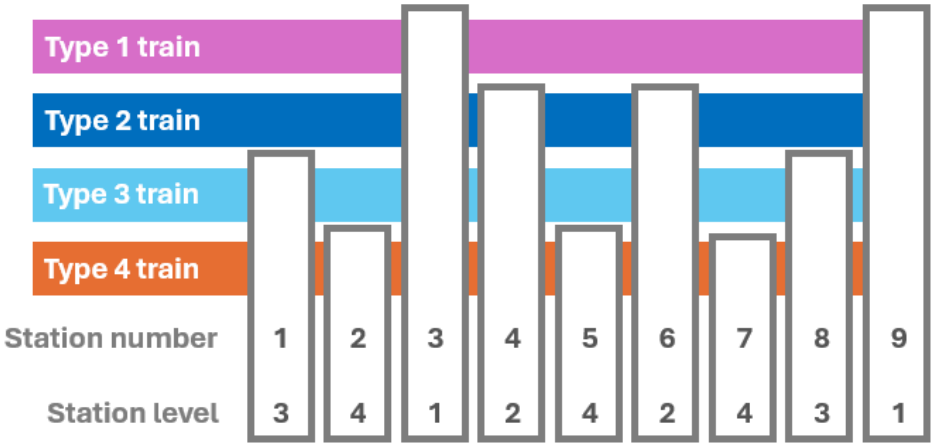


Figure 1: An example of 4 train types. In this example, a train of type 2 stops at stations 3, 4, 6 and 9 in order.

The time to travel between any two consecutive stops on any train's route is exactly 1 minute. Passengers can also switch between different train types at any station. These transfers are instantaneous, with no waiting time. In the example below, travelling from station 2 to station 7 takes 4 minutes.

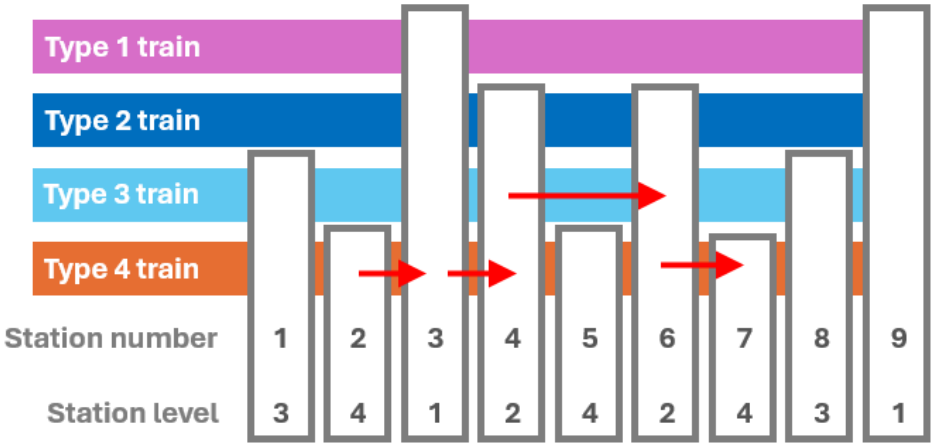


Figure 2: Travelling from station 2 to station 7.

Now, Bob wants to test the effectiveness of his proposal. He has collected data for Q potential passenger trips. For each trip i, from a starting station U_i to a destination station V_i, your task is to calculate the minimum possible travel time under his proposal.

Input

The first line of the input contains three integers N, M and Q ($2 \leq N \leq 10^5, 1 \leq M \leq N, 1 \leq Q \leq 10^5$), representing the number of stations, the number of train types and the number of trips to test.

The second line of the input contains N integers A_i ($1 \leq A_i \leq M$), representing the levels of the stations.

The next Q lines of the input contain two integers U_i and V_i ($1 \leq U_i < V_i \leq N$), representing the starting and destination stations for a trip.

Output

For each of the Q trips, output a single line containing the minimum travel time in minutes from station U_i to station V_i.

Example

input

```
9 4 5
3 4 1 2 4 2 4 3 1
2 7
3 8
1 9
2 8
2 5
```

output

```
4
3
2
4
3
```

Note

The path with the minimum travel time for the first trip is shown in the statement.

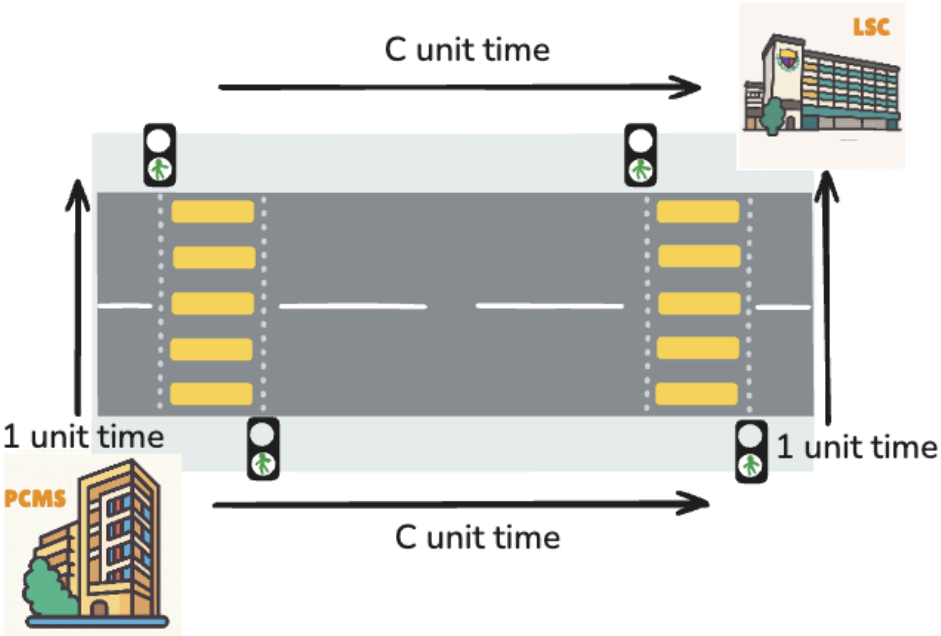
F Faster Route

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Alice and Bob want to travel from Pui Ching Middle School (PCMS) to La Salle College (LSC).

In this world, PCMS and LSC are located on opposite sides of the same road. PCMS is located at the start of the road, while LSC is located at the end of the road. There are two pedestrian crossings with traffic lights in front of both schools. Both traffic lights just turned from green to red at time 0.

Refer to the following figure for the layout.



It takes 1 unit time to cross the road, and takes C unit time to walk from the start of the road to the end of the road. The traffic light of the pedestrian crossing in front of PCMS operates in a **period** of A unit time. It means that the traffic light would be red for A unit time, then green for A unit time, and so on, alternating every A units.

Formally, you may cross the road if you arrive at the crossing in front of PCMS in time A, $A+1$, ..., $2A-1$, $3A$, $3A+1$, ..., $4A-1$, ...

Similarly, the traffic light of the pedestrian crossing in front of LSC operates in a **period** of B unit time.

Alice would take the pedestrian crossing in front of PCMS to cross the road first, then travel along the road to LSC, while Bob would travel along the road to LSC first, then take the pedestrian crossing in front of LSC.

Find the number of pairs (u, v) where $0 \le u \le U$, $0 \le v \le V$ and satisfies the following conditions:

- Alice starts at time u.
- Bob starts at time v.
- The person who starts strictly later, arrives at LSC strictly earlier.

Input

Each input contains multiple test cases. The first line contains the number of test cases T ($1 \le T \le 100$). The description of the test cases follows.

The first and only line of each test case contains five integers A, B, C, U, V ($1 \le A, B \le 10^5$, $1 \le C, U, V \le 10^9$).

It is guaranteed that the sum of $A + B$ over all test cases does not exceed 2×10^5.

Output

For each test case, output a single integer on a new line: the number of pairs (u, v) that satisfy the conditions.

Examples

input

```
3
3 2 2 5 5
314 159 265 358 979
2025 20 25 20252025 20262026
```

output

```
2
49141
10221041660
```

input

```
5
5 145 990758033 183587786 177096537
1 1274 114817991 539087542 366293156
529 2 42793654 337687912 536236694
2 465 57138389 522674854 135110507
172 7 829280391 447601408 938200233
```

output

```
6291814931
116390044533
44532871738
15622352866
18970828916
```

Note

In the first test case of Sample 1, the only two satisfied (u, v) are $(0, 1)$ and $(3, 2)$.

When $u = 0$, $v = 1$:

- Alice starts at time 0.
- Alice waits until time 3 to use the crossing, and arrives at the opposite side at time 4.
- Alice walks down the street and arrives at LSC at time 6.
- Bob starts at time 1.
- Bob walks down the street and arrives at the crossing in front of LSC at time 3.
- Bob immediately uses the crossing, arrives at the LSC at time 4.
- Bob starts strictly later than Alice, but arrives at LSC strictly earlier.

When $u = 3$, $v = 2$:

- Alice starts at time 3.
- Alice immediately uses the crossing, arrives at the opposite side at time 4
- Alice walks down the street and arrives at LSC at time 6.
- Bob starts at time 2.
- Bob walks down the street and arrives at the crossing in front of LSC at time 4.
- Bob waits until time 6 to use the crossing, and arrives at LSC at time 7.
- Alice starts strictly later than Bob, but arrives at LSC strictly earlier.

All other (u, v) where $0 \le u \le 5, 0 \le v \le 5$ do not satisfy the third condition.

G Greedy Merchants

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

In a market, there are N merchants numbered from 1 to N and N items numbered from 1 to N. Each merchant possesses an item. Initially, merchant i owns item i.

Each item has a **value** V_i and a **cost** C_i. A merchant who owns item i is **willing to trade** it for another item j if the value of item j is greater than or equal to the cost of item i. A trade between two merchants is **fair** if both merchants involved are willing to trade their current item for another. Formally, a trade between the merchant who owns item i and the merchant who owns item j is fair if and only if $V_i \geq C_j$ and $V_j \geq C_i$. Upon a fair trade, the two merchants swap their items, while the values and costs of the items remain unchanged. Note that the value of an item may not necessarily be greater than its cost.

There are Q independent queries from the merchants. In the i-th query, merchant u, who initially owns item u, wishes to obtain item v through a sequence of fair trades. Hence, he would like to know whether it is possible for him to obtain item v and, if so, the minimum number of trades required. Note that trades can occur between any pair of merchants, neither merchant u nor merchant v are necessarily involved in the trades. Since the queries are made before any trades begin, each query should be considered independently based on the initial state of the market.

Write a program to answer the queries made by the merchants.

Input

The first line contains an integer N ($2 \leq N \leq 2 \times 10^5$), the number of merchants.

The second line contains N integers V_i ($1 \leq V_i \leq 10^9$), the value of the i-th item.

The third line contains N integers C_i ($1 \leq C_i \leq 10^9$), the cost of the i-th item.

The fourth line contains an integer Q ($1 \leq Q \leq 2 \times 10^5$), the number of queries.

The next Q lines contain the descriptions of the queries. The i-th of these lines contains two integers, u and v ($1 \leq u, v \leq N$). It is guaranteed that $u \neq v$.

Output

For each query, output a single line. If it is possible for merchant u to obtain item v, output the minimum number of trades required. Otherwise, output `-1`.

Examples

input

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

output

```
2
1
-1
```

input

```
5
9 3 7 10 5
4 1 3 6 9
2
1 2
2 5
```

output

```
2
3
```

Note

In Sample 1, there are 5 merchants and 3 queries.

In the first query, merchant 2 wishes to obtain item 1. He can do so by first trading with merchant 3, then trading with merchant 1. Note that merchant 2 cannot directly trade with merchant 1 since $V_2 < C_1$. Hence, 2 is the minimum number of trades required.

In the second query, merchant 1 can directly trade with merchant 4 to obtain item 4. The minimum number of trades required is hence 1.

In the third query, it can be proven that it is impossible for merchant 3 to obtain item 5 no matter how many trades are made.

H Hurricane Signal

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

The Heung Shing Observatory has recently discovered a new hurricane approaching Heung Shing!

The world can be viewed as an infinite 2D Cartesian plane, and Heung Shing is located at coordinates (H_x, H_y). The hurricane is currently located at coordinates (C_x, C_y). It is predicted to move along a straight line that forms an angle of D degrees counterclockwise with the positive x-axis, however, the angle formed by the actual trajectory may be different due to a multitude of factors – it can be any real value between $D - U$ and $D + U$ degrees with uniform probability. When the hurricane comes within a 1 unit radius (inclusive) of Heung Shing, measured in Euclidean distance, a Hurricane Signal is issued.

Consider the two sample scenarios shown in Figures 1 and 2. In both figures, Heung Shing is at point H and the hurricane is currently at point C. The predicted trajectory of the hurricane is indicated by the green arrow. The actual trajectory of the hurricane must be within the sector formed by the red lines.

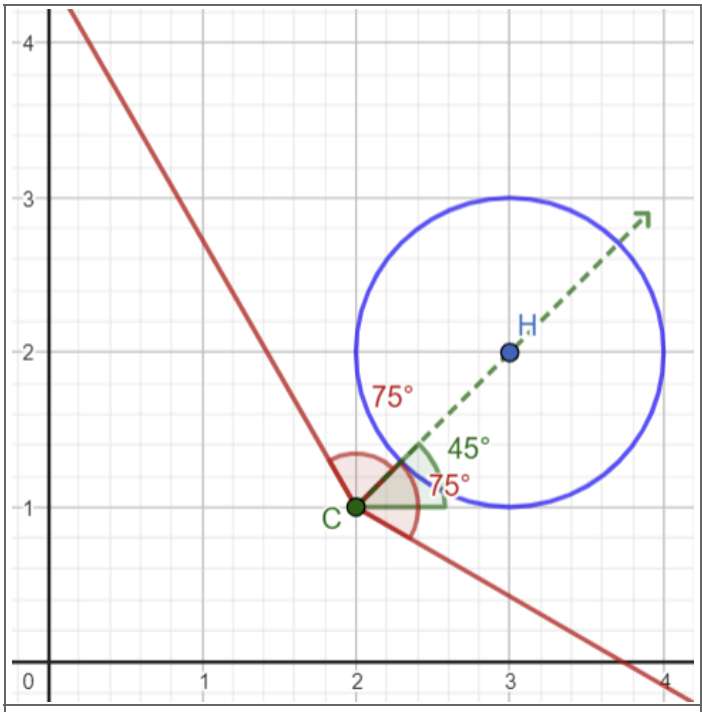


Figure 1:

$H = (3, 2), C = (2, 1), D = 45°, U = 75°$

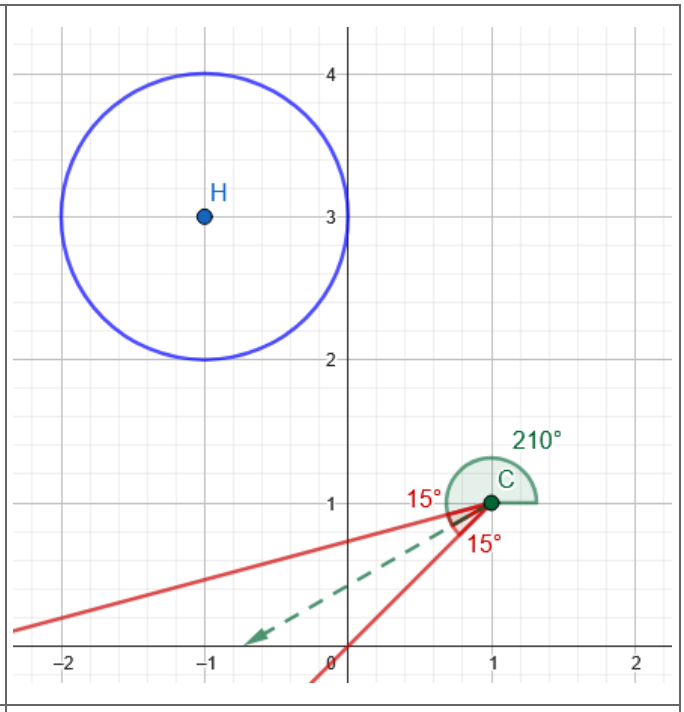


Figure 2:

$H = (-1, 3), C = (1, 1), D = 210°, U = 15°$

A Hurricane Signal is issued when the hurricane enters the blue circle. In Figure 1, the Hurricane Signal may or may not be issued in the future depending on its actual trajectory. In Figure 2, the Hurricane Signal is never issued.

Charlie is now interested in finding the probability that the hurricane will trigger a Hurricane Signal, either at its current position or somewhere along its future trajectory. Help Charlie find the answer!

Input

The first line consists of two integers H_x and H_y ($-1000 \leq H_x, H_y \leq 1000$), the coordinates of Heung Shing.

The second line consists of two integers C_x and C_y ($-1000 \leq C_x, C_y \leq 1000$), the coordinates of the current position of the hurricane.

The third line consists of an integer D ($0 \leq D < 360$) and an integer U ($0 < U < 90$), the angle (in degrees, counterclockwise) formed by the predicted trajectory and the positive x-axis, and the maximum deviation (in degrees) of the prediction.

Output

Output a decimal number, the probability of a Hurricane Signal being issued. Your answer is considered correct if the absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be x and the correct answer be y. Your answer is accepted if and only if $\frac{|x-y|}{\max(1,|y|)} \leq 10^{-6}$.

Examples

input	output
3 2 2 1 45 75	0.6
-1 3 2 1 210 15	0
-1 3 4 -2 120 12	0.213754265

Note

See Figure 1 for an illustration of the first sample. The angle of the hurricane's path relative to the positive x-axis can be any value between -30° and 120° with uniform probability. Heung Shing will issue the Hurricane Signal if this angle is between 0° and 90°. Therefore the required probability is $\frac{90}{150} = 0.6$.

See Figure 2 for an illustration of the second sample. The angle of the hurricane's path relative to the positive x-axis can be any value between 195° and 225° with equal probability. The hurricane will always be more than 1 unit away from Heung Shing.

I Interesting PuiLaTiu

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Today, in the year 2025, marks the 9th Pui Ching-La Salle Programming Challenge (referred to as PuiLaTiu below) ever held! PuiLaTiu was first held in 2016 and paused for a year in 2020 due to COVID restrictions (and hence the 5th PuiLaTiu was held in 2021).

Alice is fascinated because both the edition number (9) and the year (2025) of this PuiLaTiu are perfect squares. She defines an *interesting* PuiLaTiu as one where both the edition number and the year it is held are perfect squares. She wonders: assuming PuiLaTius will be held once every year in the future, how many PuiLaTius held in years L to R (inclusive) are interesting?

Alice is distracted thinking about this during the contest. As Alice's teammate, you should answer her question to keep her from being distracted.

Input

The input consists of two integers L and R ($2021 \leq L \leq R \leq 10^9$) as indicated in the statement.

Output

Output a single integer: the number of interesting PuiLaTius in years L to R (inclusive).

Examples

input	output
2021 2025	1

input	output
2026 2030	0

input	output
2100 2199	1

Note

For Sample 1, the only interesting PuiLaTiu is held in 2025 (this year)!

For Sample 2, the 10-th PuiLaTiu is held in 2026, the 11-th PuiLaTiu is held in 2027, and so on. It can be proven that there are no interesting PuiLaTius in the next 5 years.

For Sample 3, the only interesting PuiLaTiu in the 22nd century is the 100-th PuiLaTiu held in the year 2116, with $100 = 10^2$ and $2116 = 46^2$.

This page is intentionally left blank.

J Jumping Game

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Froggy the frog is playing a jumping game. In the game, there are 1 **starting platform**, followed by N **jumping platforms**, numbered from 1 to N, arranged in a line. The starting platform is 1 block wide and the i-th jumping platform is A_i blocks wide, where A_i is always an odd number. For each jumping platform, the exact middle block, which is the $\lceil \frac{A_i}{2} \rceil$-th block, is a scoring block. The distance between the starting platform and the 1-st jumping platform is D_1 blocks, and the distance between the $(i-1)$-th jumping platform and the i-th jumping platform is D_i blocks for $2 \le i \le N$.

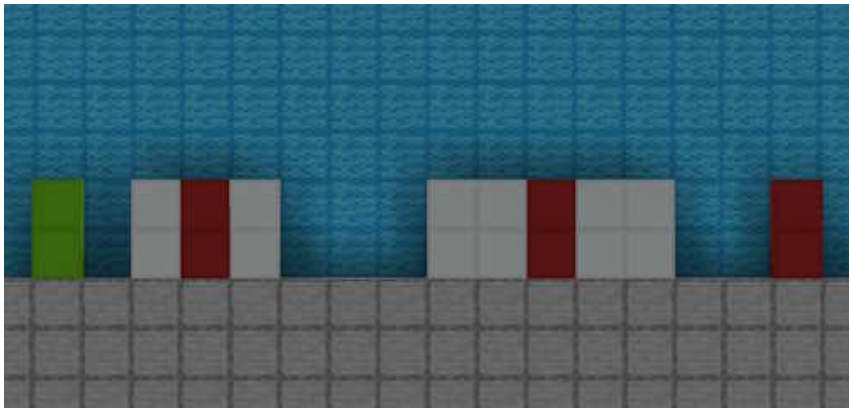


Figure 1: An example where $N = 3$, $A = [3, 5, 1]$ and $D = [1, 3, 2]$. The starting platform is green, and the scoring blocks are red.

Froggy starts by standing on the starting platform, and his goal is to reach the N-th jumping platform by performing exactly N jumps, landing on each platform exactly once. If he successfully lands all N jumps, he completes the game. Additionally, he can jump at most K_i blocks for his i-th jump.

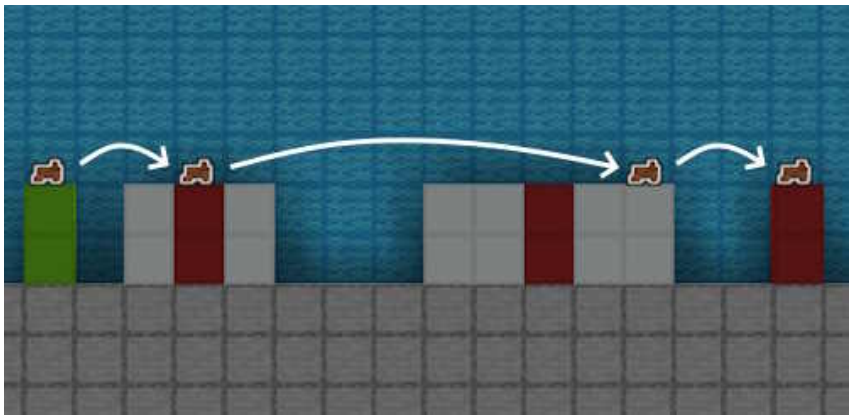


Figure 2: For $K = [4, 10, 3]$, a valid jump sequence could have distances of $3, 9, 3$ blocks.

Froggy can also earn points by landing on scoring blocks. For each scoring block he lands on, he will earn 1 point. However, he gets points only if he completes the game. In the example above, he completes the game with 2 points, which is the maximum achievable score.

Your task is to determine if Froggy can complete the game, and if so, determine the maximum possible score he can achieve.

Input

The first line of the input contains one integer N ($1 \leq N \leq 2 \times 10^5$), representing the number of jumping platforms.

The second line of the input contains N integers A_i ($1 \leq A_i < 10^9, A_i$ is odd), representing the width of each jumping platform.

The third line of the input contains N integers D_i ($1 \leq D_i \leq 10^9$), representing the distance between each jumping platform and its previous platform.

The fourth line of the input contains N integers K_i ($1 \leq K_i \leq 10^9$), representing the maximum number of blocks Froggy can jump for each jump.

Output

If it is impossible to complete the game, you should print `-1`.

Otherwise, you should print the maximum achievable score on a single line.

Examples

input

```
3
3 5 1
1 3 2
4 10 3
```

output

```
2
```

input

```
2
11 5
3 3
5 5
```

output

```
-1
```

input

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

output

```
0
```

Note

In Sample 1, the platforms and Froggy's moves are shown in Figures 1 and 2 above.

K Keep Sorting

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE
第九屆·2025年8月13日

In Linear Sorting Process Computational Programming City (LSPCPC), a crucial component of the city's data systems is the **Linear-Sorting Machine**. This machine is designed to sort permutations of length $3N$.

A **permutation** of length $3N$ is an array consisting of $3N$ distinct integers from 1 to $3N$ in arbitrary order. For example, $[6, 2, 5, 1, 3, 4]$ is a permutation of length 6 ($N = 2$), but $[1, 3, 3]$ is not a permutation.

Disaster struck when yesterday's hurricane caused severe flooding, damaging the machine's circuitry. While still operational, the machine's capabilities are now limited. It can only sort continuous segments of **at most** $2N$ **numbers** per operation.

An operation can be defined by a pair of integers L and R, which sorts the subarray $A_L, A_{L+1} \dots A_R$ in ascending order. As the segment can contain at most $2N$ numbers, for any operation (L, R), the condition $R - L + 1 \leq 2N$ must hold.

The Mayor has declared a state of emergency as the city's data arrays have become scrambled. You've been tasked with sorting a critical permutation of $3N$ integers that was being processed when the disaster hit, returning it to perfect ascending order (i.e., $A_i = i$ for all $1 \leq i \leq 3N$). Your goal is to find a sequence of operations to sort the permutation and determine the minimum number of sorting operations required to achieve this.

Input

The first line contains a single integer N ($1 \leq N \leq 10^5$).

The second line contains $3N$ integers, $A_1, A_2, \dots, A_{3N}$, representing the permutation that needs to be sorted.

Output

The first line should contain an integer K, representing the minimum number of operations required to sort the permutation.

The following K lines should contain the sequence of operations that attain the minimum number of operations.

Each line should contain a pair of integers L_i and R_i, representing the i-th operation in chronological order.

If multiple solutions achieve the minimum number of operations, you may output any one of them.

Examples

input	output
2 1 4 3 2 5 6	1 2 4
3 1 3 2 4 5 6 7 9 8	2 2 3 8 9

Note

In the first sample,

- The original array is $[1, 4, 3, 2, 5, 6]$.
- After applying the operation $(2, 4)$, it becomes $[1, 2, 3, 4, 5, 6]$, which is sorted in ascending order.
- It can be proven that 1 operation is minimal.

In the second sample,

- The original array is $[1, 3, 2, 4, 5, 6, 7, 9, 8]$.
- After applying the first operation $(2, 3)$, it becomes $[1, 2, 3, 4, 5, 6, 7, 9, 8]$.
- After applying the second operation $(8, 9)$, it becomes $[1, 2, 3, 4, 5, 6, 7, 8, 9]$, which is sorted in ascending order.
- It can be proven that 2 operations are minimal.

L Linesweeper

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Alice is an amateur Linesweeper player who is learning how to play the game!

In Linesweeper, mines are randomly placed on a board that consists of a single row with N cells, numbered from 1 to N from left to right. Some cells contain mines while some cells do not contain mines.

Initially, all cells are unopened, and no information will be displayed. In each move, a player selects an unopened cell to open it. If the chosen cell contains a mine, the player loses immediately. Otherwise, a number is displayed in the opened cell, indicating how many adjacent cells contain mines. Two cells are considered adjacent if they share a side. As each cell has at most two adjacent cells, the number displayed in an opened cell will be `0`, `1`, or `2`.

Alice has not lost yet. Thus, all opened cells do not contain mines. However, she is struggling to solve her Linesweeper; therefore, she shares the current board and seeks your help. Can you determine how many unopened cells she can open in her next move that are guaranteed not to contain mines?

Input

The first line consists of a single integer, N ($1 \leq N \leq 10^5$), the number of cells.

The second line contains a string of length N, consisting only of the characters `0`, `1`, `2`, and `?`. If the i-th character is `?`, the i-th cell is unopened. If the i-th character is `0`, `1`, or `2`, it represents the number of mines adjacent to the i-th cell.

The information provided is guaranteed to be consistent and all opened cells do not contain mines.

Output

Output a single integer: the number of unopened cells that are guaranteed not to contain mines.

Examples

input

```
3
?1?
```

output

```
0
```

input

```
7
0??2?1?
```

output

```
2
```

input

```
4
?11?
```

output

```
0
```

Note

In Sample 1, we can only determine that either the 1-st or 3-rd cell contains a mine. However, we cannot deduce exactly which one contains a mine. Therefore, there are no unopened cells that are guaranteed not to contain mines.

In Sample 2, since there is no mine next to the 1-st cell, we can deduce that the 2-nd cell does not contain a mine. In the 4-th cell, there should be 2 mines adjacent to it; therefore, we can deduce that the 3-rd and 5-th cells contain mines. Since the 5-th cell contains a mine and there is only 1 mine next to the 6-th cell, we can deduce that the 7-th cell does not contain a mine. Therefore, the 2-nd and 7-th cells are guaranteed not to contain mines.

In Sample 3, we can deduce that the 1-st and 4-th cells contain mines. Therefore, there are no unopened cells that are guaranteed not to contain mines.