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

10TH · 2026

JUL 25, 2026 (SATURDAY)

LA SALLE COLLEGE

ID	NAME	TIME LIMIT	MEMORY LIMIT
A	Anniversary	0.5 seconds	256 MB
B	Burnside's Lemma	0.5 seconds	256 MB
C	Circle Quiz Relay	0.5 seconds	256 MB
D	Doomsday of the Disguisers	0.5 seconds	256 MB
E	Emoji Riddle	1 second	256 MB
F	Forest Guardian Traversal	0.5 seconds	256 MB
G	Go Shoot!	0.5 seconds	256 MB
H	Hollow Knight: Silksong	0.5 seconds	256 MB
I	Ice Cream Sort	0.5 seconds	256 MB
J	Jettisoned Joins	0.5 seconds	256 MB
K	KMP Evasion	0.5 seconds	256 MB
L	Last Order	0.5 seconds	256 MB

C++ `g++-7 -static -std=c++17 -Wno-unused-result -DEVAL -lm -s -O2`

Python 3 `python3.5 -O -S`

Compilation Time Limit: 10 seconds Memory Limit: 512 MB

AUTHORS / PREPARED BY

Cheng Hei Chit, Cheng Yu San, Choy Tsz Chai Berton, Kwok Ching Him,
Lai Hong Tung Honrich, Lu Yi Fung, Lung Sui Hei, Mak Chun Hin Ivan,
Wong Cheuk Kiu, Wong Chun, Wong Ho Yan, Wong Man Hang, Wong Tsz Him,
Yeung Man Tsung, Yeung Shing Hei Angus, Yuen Lok Kan Ethen

NOTES

1. Output strings are checked case-sensitively.
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. `long long` for C/C++) may be required for some problems.

A Anniversary

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

La Salle – Pui Ching Programming Challenge is celebrating its 10th anniversary. To mark the occasion, the organizers want to build a large `10` out of coloured balloons, and no two balloons that touch each other may share the same colour.

The `10` is drawn on a grid with N rows and $M+2$ columns, where rows are numbered 1 to N from top to bottom and columns are numbered 1 to $M+2$ from left to right. The cell in row r and column c is denoted (r, c). A cell containing a balloon is called a *balloon cell*; all other cells are called *empty cells*.

The following figure shows the grid for $N = 5$ and $M = 3$, where `#` denotes a *balloon cell* and `.` denotes an *empty cell*.

```
#.###
#.#.#
#.#.#
#.#.#
#.###
```

The grid is divided into three parts:

- The digit `1`: column 1, every cell $(r, 1)$ for $1 \le r \le N$ contains a balloon.
- The gap: column 2 contains no balloons, for every row.
- The digit `0`: columns 3 to $M+2$, forming an $N \times M$ rectangle. A cell in this rectangle contains a balloon if and only if it lies on the boundary of the rectangle, that is, it is in row 1, row N, column 3, or column $M+2$. All other cells in this rectangle contain no balloon.

There are K colours of balloons available, numbered 1 to K. There are exactly A_i balloons of colour i. Every balloon used in the construction must be assigned one of the K colours, and a colour may only be used up to A_i times in total. Two cells are *adjacent* if they share a common edge, that is, (r, c) is adjacent to $(r-1, c)$, $(r+1, c)$, $(r, c-1)$, and $(r, c+1)$, if any exist. Two balloon cells that are *adjacent* must **not** be assigned the same colour.

Determine whether it is possible to assign a colour to every balloon cell so that these conditions are satisfied, and if so, output one such assignment.

Input

The first line contains three integers N, M, and K ($3 \le N, M \le 100$, $1 \le K \le 5$), the height of the digit `1`, the width of the digit `0`, and the number of colours. The second line contains K integers $A_1, A_2, \ldots, A_K$ ($0 \le A_i \le 3N + 2M - 4$), where A_i is the number of balloons of colour i.

It is guaranteed that $A_1 + A_2 + \cdots + A_K = 3N + 2M - 4$, the exact number of balloon cells in the `10` (that is, N balloon cells for the digit `1` plus $2N + 2M - 4$ balloon cells for the digit `0`). In other words, every balloon must be used.

Output

The first line should contain the string `POSSIBLE` if a valid assignment exists, or `IMPOSSIBLE` otherwise. If the first line is `POSSIBLE`, output N additional lines, each consisting of exactly $M+2$ characters, describing the coloured grid. The j-th character of the i-th line must be:

- a digit from `1` to `5` representing the colour, if cell (i, j) is a *balloon cell*,
- `.`, if cell (i, j) is an *empty cell*.

If the first line is `IMPOSSIBLE`, do not output anything else.

Examples

input

```
5 3 3
6 6 5
```

output

```
POSSIBLE
1.123
2.3.1
1.2.2
2.1.3
3.321
```

input

```
3 3 2
6 5
```

output

```
POSSIBLE
1.121
2.2.2
1.121
```

input

```
3 3 2
9 2
```

output

```
IMPOSSIBLE
```

Note

In the first sample, the sample output uses colours $1, 2, 1, 2, 3$ for the digit `1` and uses $1, 2, 3$ around the digit `0` perimeter, using exactly 6, 6, and 5 balloons of each colour, with no two adjacent balloons sharing a colour.

In the second sample, the sample output uses colours $1, 2, 1$ for the digit `1` and uses $1, 2$ around the digit `0` perimeter, using exactly 6 and 5 balloons of each colour, with no two adjacent balloons sharing a colour.

In the third sample, with only 2 colours, the digit `1` must use one colour twice and the other once, while the even 8-cell digit `0` cycle must alternate colours exactly 4-4. Since colour 2 has only 2 balloons, no split satisfies both requirements, so the answer is `IMPOSSIBLE`.

B Burnside's Lemma

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

Bob is a professional chef at a five-star restaurant who specializes in cooking steaks. To avoid burning the steak, he regularly flips and rotates the steaks.

Today, Bob is sick, and he is giving you, his favorite apprentice, instructions remotely on how to flip and rotate the steaks. Bob will give you N instructions of the following format:

- `H`: you should flip the steak around the horizontal axis.
- `V`: you should flip the steak around the vertical axis.
- `C`: a positive integer X less than 360 is input on the next line, you should rotate the steak X degrees clockwise.
- `A`: a positive integer X less than 360 is input on the next line, you should rotate the steak X degrees anti-clockwise.

Note that all flips and rotations are performed relative to your fixed perspective looking down at the frying pan, not the steak's local orientation.

In order to make sure you followed his instructions correctly, after the N instructions, Bob wants you to report the final state of the steak, whether or not it is flipped, and how many degrees it is rotated **clockwise**, with respect to its initial state.

Input

The first line contains one integer N ($1 \leq N \leq 10^5$), representing the number of instructions Bob gave you.

The following lines represent Bob's instructions, as mentioned above.

Output

The output should consist of two lines. The first line should contain `Flipped` or `Not flipped` to indicate whether the steak is flipped at last.

The second line should contain a non-negative integer less than 360, indicating how many degrees the steak is rotated at last.

Examples

input

```
4
C
45
H
A
15
V
```

output

```
Not flipped
240
```

input

```
1
H
```

output

```
Flipped
180
```

input

```
2
H
V
```

output

```
Not flipped
180
```

input

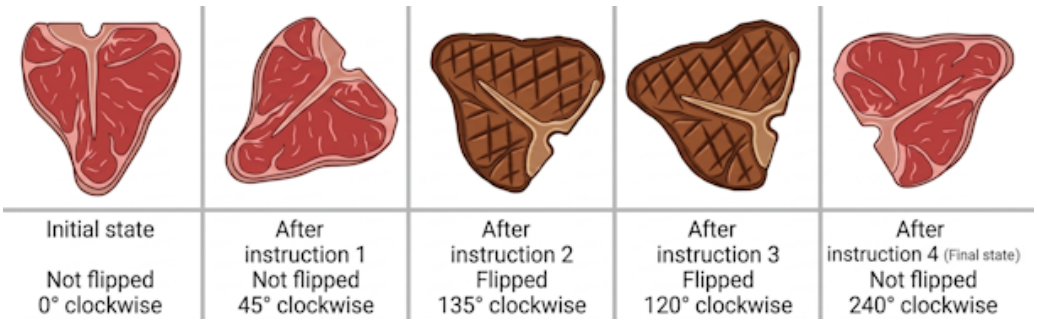
```
4
C
270
C
180
A
120
H
```

output

```
Flipped
210
```

Note

The following image shows the steak after each instruction in the first sample test.



C Circle Quiz Relay

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

There are N students sitting in a circle, numbered 1 through N clockwise. For every $1 \leq i < N$, student i is sitting to the right of student $i+1$. Student N is sitting to the right of student 1.

Every student has exactly one best friend. The best friend of student i is student P_i. Note that this is a one-way relationship: the best friend of student P_i is not necessarily student i. Every student is a best friend of exactly one student, and a student is allowed to be their own best friend. Formally, P is a permutation of 1 to N.

The teacher prepares N questions for the class. The N questions must be answered by N distinct students. Every time a student completes answering a question, the teacher tells them to choose a student to answer the next question. The student always chooses their best friend. If the student chosen for the next question has already answered a question before, the teacher says **"you have already answered before"**, and tells the chosen student to choose another student, in which case the chosen student always chooses the student sitting immediately to their left. This process continues until the chosen student has not answered a question before. The class ends when all N questions have been answered.

Since time is running short, the teacher wants to minimize the number of times he says **"you have already answered before"**. The teacher knows the best friend of every student, and can pick any student to be the first one to answer a question. Write a program to find the minimum possible number of times the teacher must say **"you have already answered before"** if the teacher chooses the first student optimally.

Input

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

The second line of the input contains N integers. The i-th integer is P_i $(1 \leq P_i \leq N)$, representing the best friend of student i. It is guaranteed that P is a permutation of 1 to N.

Output

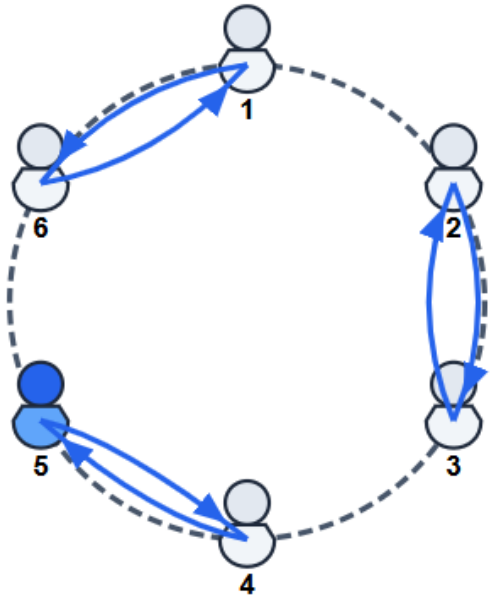
Output a single integer on the first line: the minimum possible number of times the teacher must say **"you have already answered before"** if the teacher chooses the first student optimally.

Examples

input	output
6 6 3 2 5 4 1	3
5 1 2 3 4 5	4
4 2 3 4 1	0

Note

The following figure illustrates the seating arrangement for the students in Sample 1. The arrows illustrate the best-friend relationships between the students.



Assume the teacher picks student 5 as the first student to answer a question. Then, the following happens:

1. Student 5 answers a question, and then chooses their best friend, student 4.
2. Student 4 answers a question, and then chooses their best friend, student 5.
3. Student 5 has already answered a question. So, the teacher says **"you have already answered before"**. Then, student 5 chooses the student sitting immediately to their left, student 6.
4. Student 6 answers a question, and then chooses their best friend, student 1.
5. Student 1 answers a question, and then chooses their best friend, student 6.
6. Student 6 has already answered a question. So, the teacher says **"you have already answered before"**. Then, student 6 chooses the student sitting immediately to their left, student 1.
7. Student 1 has already answered a question. So, the teacher says **"you have already answered before"**. Then, student 1 chooses the student sitting immediately to their left, student 2.
8. Student 2 answers a question, and then chooses their best friend, student 3.
9. Student 3 answers a question. Then, all questions are answered and the class ends.

The teacher has said **"you have already answered before"** 3 times. Note that there are other ways to pick the first student that result in the same number of times, but the teacher must say **"you have already answered before"** at least 3 times.

D Doomsday of the Disguisers

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

Benson is playing a game called *Meccha Chameleon* as a hunter. In this game, a hunter is required to find out the camouflaged chameleons and kill them with bullets.

After detailed investigation, he finds out that all the chameleons are hiding in an $N \times M$ grid, with each cell (numbered from $(1, 1)$ to (N, M)) containing exactly 1 chameleon (so there are in total $N \times M$ chameleons). Here, (r, c) represents the cell at the r-th row and c-th column.

He decides to become a cold, heartless killer by executing Q killing operations in order.

In each operation, he chooses a row or a column. If he chooses row x, he stands at $(x, 0)$ and shoots 1 bullet for each chameleon still alive in that row. Each bullet passes through $(x, 1)$, $(x, 2)\ldots$ until it hits and kills a chameleon. If he chooses column y, he stands at $(0, y)$ and shoots 1 bullet for each chameleon still alive in that column. Each bullet passes through $(1, y)$, $(2, y)\ldots$ until it hits and kills a chameleon.

Unfortunately, he loses track of how many chameleons in total he has killed. Luckily, he still has a record of N, M, Q and the operations he has executed (which are guaranteed to be distinct, Benson is not that stupid to apply the same operation twice or more!!), so he decides to hand you all the recorded data, and assign this task to you, a genius competitive programmer.

Input

The first line contains 3 integers N, M, Q $(1 \leq N, M, Q \leq 100)$.

The following Q lines each contain a character T, and an integer X. T is either `R` or `C`.

If T is `R`, this indicates that Benson chooses row X. $(1 \leq X \leq N)$

If T is `C`, this indicates that Benson chooses column X. $(1 \leq X \leq M)$

It is guaranteed that the Q lines are distinct.

Output

Output an integer, the number of chameleons Benson has killed.

Examples

input

```
3 4 3
R 1
R 3
C 2
```

output

```
9
```

input

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

output

```
10
```

This page is intentionally left blank.

E Emoji Riddle

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

Alice is designing a riddle where emojis are used as clues for words. For example, the emojis snake and crown may separately suggest "snake" and "crown", but together form a phrase meaning "slacking" (being lazy at work) in Hong Kong slang.

To study how different riddles behave using NLP methods, Alice represents each word and each emoji by a 2D feature vector.

There are N words, numbered from 1 to N. Word i has feature vector (a_i, b_i). There are also M emojis, numbered from 1 to M. Emoji j has feature vector $(t_j, 1)$.

The **relevance** between word i and emoji j is defined as the dot product of their feature vectors, $a_i t_j + b_i$. The word suggested by an emoji is the word with the largest relevance to it.

An emoji riddle consists of two different emojis x and y. The relevance between word i and the riddle is the **sum** of the two individual relevance values, that is, $(a_i t_x + b_i) + (a_i t_y + b_i)$. The word suggested by a riddle is the word with the largest relevance to the riddle.

Alice calls a riddle **surprising** if the two emojis individually suggest two different words, but the riddle itself suggests a third word, different from both of them.

Your task is to count the number of unordered pairs of different emojis that form a surprising riddle.

It is guaranteed that there are no ties in any maximum that needs to be considered. More precisely, for every pair of different emojis x and y: there is exactly one word with the largest relevance to emoji x, exactly one word with the largest relevance to emoji y, and exactly one word with the largest relevance to the riddle formed by x and y.

Input

The first line contains two integers N and M ($3 \le N \le 10^5, 2 \le M \le 10^5$), representing the number of words and the number of emojis.

The next N lines each contain two integers a_i and b_i ($-10^9 \le a_i, b_i \le 10^9$), the feature vector of word i.

The last line contains M integers $t_1, t_2, \ldots, t_M$ ($-10^9 \le t_i \le 10^9$). The feature vector of emoji i is $(t_i, 1)$.

Output

Output a single integer: the number of surprising riddles.

Examples

input

```
3 3
0 0
1 0
2 -10
-5 7 17
```

output

```
1
```

input

```
3 4
0 0
1 0
2 -10
-9 -5 15 19
```

output

```
4
```

input

```
10 10
6 93
-7 198
-10 -28
-5 -192
12 -104
2 100
-3 -155
-13 121
-1 -14
9 -194
-53 -31 -15 -9 5 9 33 46 55 57
```

output

```
22
```

Note

For Sample 1, the relevance values between each word and each emoji are:

	word 1 $(0, 0)$	word 2 $(1, 0)$	word 3 $(2, -10)$
emoji 1 $(-5, 1)$	0	-5	-20
emoji 2 $(7, 1)$	0	7	4
emoji 3 $(17, 1)$	0	17	24

Therefore emoji 1, emoji 2, and emoji 3 suggest word 1, word 2, and word 3 respectively.

Consider the riddle formed by emoji 1 and emoji 3. The relevance values of the riddle are obtained by adding the first and third rows:

	word 1	word 2	word 3
riddle $(1, 3)$	0	12	4

So even though the two emojis individually suggest word 1 and word 3, their riddle suggests word 2. This is the only surprising riddle in Sample 1.

In Sample 2, there are four surprising riddles: $(1, 3)$, $(1, 4)$, $(2, 3)$, and $(2, 4)$.

F Forest Guardian Traversal

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

Figure 1: Brr Brr Patapim

Brr Brr Patapim is the mythical guardian of an ancient forest of N nodes. A forest is a graph in which all connected components are trees. An undirected connected graph without cycles and self-loops is called a tree. To ensure the safety of the woods, he performs a routine inspection by traversing the forest using a randomized Depth-First Search (DFS) approach governed by the following rules:

1. If it is the first step, or if he has completely visited all the nodes in the current tree component, he will uniformly at random pick a node in the entire forest that has not yet been visited. He immediately moves to that node.
2. If the current node has at least one neighbor that has not been visited yet, he will uniformly at random choose one of these unvisited neighbors and move to it.
3. If all neighbors of the current node have already been visited, but there are still unvisited nodes remaining in the current tree component, he backtracks to the node he came from when he first visited the current node, and continues exploration from there.

Every time Brr Brr Patapim steps onto a node for the **first time**, he appends it to his patrol log. Since his choices are randomized, the final patrol log is a random permutation of all N nodes.

Your task is to help the forest spirits predict his behavior. For each node from 1 to N, calculate the expected position (1-indexed order) of that node in Brr Brr Patapim's final patrol log.

It can be shown that the expected position is in the form of $\frac{P}{Q}$, where P and Q are non-negative integers and $Q \not\equiv 0 \pmod{998244353}$. Output the value of $P \cdot Q^{-1} \pmod{998244353}$.

Input

The first line contains two integers N and M ($1 \le N \le 10^5, 0 \le M < N$), representing the number of nodes and the number of edges in the forest, respectively.

Each of the following M lines contains two integers u and v ($1 \le u, v \le N, u \ne v$), representing an undirected edge between node u and node v. It is guaranteed that the given graph is a forest.

Output

The output consists of N lines, each containing a single integer. The i-th integer represents the expected position of node i in the final patrol log modulo 998244353.

Examples

input	output
3 1 1 2	831870296 831870296 332748120

input	output
9 6 1 2 2 3 3 4 2 5 6 7 7 8	241242390 341066824 840189001 540715696 241242390 790276785 291154608 790276785 915057331

Note

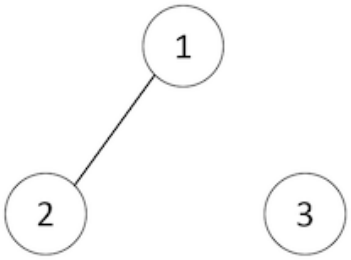


Figure 2: The forest of sample 1

In sample 1, the forest contains a 2-node tree $\{1, 2\}$ and an isolated node $\{3\}$.

- If Brr Brr Patapim starts at node 1 (probability $\frac{1}{3}$): He is forced to visit neighbor 2, then jump to 3. The only log is $[1, 2, 3]$.
- If Brr Brr Patapim starts at node 2 (probability $\frac{1}{3}$): He is forced to visit neighbor 1, then jump to 3. The only log is $[2, 1, 3]$.
- If Brr Brr Patapim starts at node 3 (probability $\frac{1}{3}$): He then picks node 1 or 2 next with equal probability, yielding logs $[3, 1, 2]$ and $[3, 2, 1]$ (each with probability $\frac{1}{6}$).

Expected position for node 1 = $\frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 2 + \frac{1}{6} \cdot 2 + \frac{1}{6} \cdot 3 = \frac{11}{6} \equiv 831870296 \pmod{998244353}$

Expected position for node 2 = $\frac{1}{3} \cdot 2 + \frac{1}{3} \cdot 1 + \frac{1}{6} \cdot 3 + \frac{1}{6} \cdot 2 = \frac{11}{6} \equiv 831870296 \pmod{998244353}$

Expected position for node 3 = $\frac{1}{3} \cdot 3 + \frac{1}{3} \cdot 3 + \frac{1}{6} \cdot 1 + \frac{1}{6} \cdot 1 = \frac{7}{3} \equiv 332748120 \pmod{998244353}$

G Go Shoot!

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

The Beyblade club at your school meets every Friday after classes. Members bring their favourite Beyblades, share their latest discoveries, and challenge one another in friendly 3-on-3 battles. At this week's meeting, you decide to build a new deck using parts available at a nearby shop.

In Beyblade X 3-on-3 battles, each player prepares a deck with exactly 3 Beyblades, each with 1 Blade, 1 Ratchet and 1 Bit. Each available part may be used at most once.

Every part has an Attack value, a Defense value, and a Stamina value. For a Beyblade, let A, D, and S be, respectively, the sums of the Attack, Defense, and Stamina values of its three parts. Its power is $\max(A, D) + S$. The power of a deck is the sum of the powers of its 3 Beyblades.

The shop you are visiting has some Beyblade X parts for sale. Each part has a cost. After buying the parts, you may assemble them into the three Beyblades in any way, provided that every Beyblade has one part of each type. The total cost of all nine selected parts must not exceed your budget X.

Suppose the shop has the following parts for sale and your budget is $X = 110$:

Consider the following deck:

- Wizard Rod 5-70 DB: $\max(15 + 24 + 15, 25 + 17 + 20) + (60 + 19 + 55) = 196$.
- Knight Fortress GV 8-70 UN: $\max(25 + 16 + 10, 55 + 20 + 60) + (20 + 24 + 20) = 199$.
- Bahamut Blitz BK 1-50 I: $\max(65 + 36 + 50, 20 + 18 + 15) + (15 + 6 + 5) = 177$.

The 9 selected parts cost 104, which is within the budget of 110. Their deck power is $196 + 199 + 177 = 572$, and no valid deck has greater power.

Now you have a different selection of parts and a different budget. Build the strongest deck possible.

Input

The first line contains an integer T $(1 \le T \le 10)$, the number of test cases.

The first line of each test case contains four integers N, M, K, and X $(3 \le N, M, K \le 200$, $1 \le X \le 2000)$, the numbers of available Blades, Ratchets, and Bits, and your budget.

The next N lines describe the Blades. Each line contains four integers c, a, d, and s $(1 \le c \le X$, $0 \le a, d, s \le 10^6)$, the cost, Attack, Defense, and Stamina values of one Blade.

The next M lines describe the Ratchets and the following K lines describe the Bits, in the same format.

Output

For each test case, output one integer: the maximum possible power of a valid deck whose total cost is at most X. Output -1 if it is impossible to construct a valid deck within the budget.

Example

input

```
2
4 5 5 110
13 35 55 10
18 65 20 15
12 15 25 60
14 25 55 20
17 26 20 14
8 24 17 19
7 36 18 6
9 16 20 24
16 34 18 8
21 40 5 10
11 15 20 55
13 50 15 5
20 50 5 5
12 10 60 20
3 3 3 9
1 0 0 0
1 0 0 0
2 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
```

output

```
572
-1
```

Note

The sample input contains two test cases ($T = 2$). In the first test case, the optimal deck is described in the problem statement, so the first output line is 572.

In the second test case, there are exactly three available parts of each type. Since each listed part may be used at most once, every available part must be used to construct a deck. Their total cost is 10, which exceeds the budget of 9; therefore, the second output line is -1.

H Hollow Knight: Silksong

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

In the highly competitive gaming industry, publishers carefully schedule their game launches to avoid clashing with massive, highly anticipated blockbusters. For instance, when an industry giant like Hollow Knight: Silksong or GTA6 drops, other studios scramble to clear the calendar for weeks before and after to ensure their own sales aren't cannibalized by the hype.

Bob is a master release strategist who manages the launch schedules for a coalition of game companies. He currently has N upcoming games pending release. Through extensive market research, Bob has assigned an expected popularity score, P_i, to each game i.

To ensure every game gets its fair share of the spotlight, Bob must assign a specific release day to each game such that the following condition is met: If game j is strictly more popular than game i (i.e., $P_j > P_i$), game i cannot be released within X days before or after game j. Launching a game requires immense logistical effort, so Bob can release at most one game per day.

Because Bob has full control over the calendar, he wants to compress the release schedule as much as possible to keep the momentum going. Your task as Bob's best friend is to determine the minimum number of days required to release all N games without violating the popularity rule.

Input

The first line of the input contains two space-separated integers, N ($1 \leq N \leq 10^5$) and X ($0 \leq X \leq 10^9$). The second line contains N space-separated integers $P_1, P_2, \ldots, P_N$ ($0 \leq P_i \leq 10^9$), representing the expected popularity scores of the N games.

Output

Output a single integer representing the minimum number of days required to release all N games.

Examples

input

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

output

```
17
```

input

```
3 8
2 4 1
```

output

```
19
```

Note

This is an optimal release schedule to assign the release days to the games in the sample above:

1. Game 1 (Popularity 3): Day 1
2. Game 3 (Popularity 3): Day 2
3. Game 2 (Popularity 4): Day 7 (As it is the most popular game, it blocks releases on Days 3–6 and Days 8–11)
4. Game 4 (Popularity 1): Day 12
5. Game 5 (Popularity 2): Day 17

It can be proven that there is no valid release schedule that takes fewer than 17 days.

I Ice Cream Sort

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
10TH · 25 JUL 2026

Despite the intense summer heat, the citizens are excited because today (25th of July) is officially National Hot Fudge Sundae Day (search after contest if you don't believe me)! To celebrate, the renowned local ice cream parlor is giving away its signature ten-scoop sundaes, topped with hot fudge and whipped cream, entirely for free.

The news has spread quickly, resulting in a long queue wrapping around the building. Currently, there are N customers waiting in a single line, and each person is holding a unique ticket number that was distributed earlier this morning. However, the parlor owner is quite strict about organization. He refuses to serve any ice cream until the customers are arranged so that their ticket numbers are in strictly increasing order from front to back.

To manage the eager crowd, the owner decides to implement a unique crowd-control method he invented, which he calls the **Ice Cream Sort**.

The sorting process works as follows:

1. He selects a fixed integer distance, D (where $1 \leq D \leq N$).
2. Using this distance D, he can swap the positions of any two customers who are exactly D spaces apart in the queue, formally, if a customer is in position i ($1 \leq i \leq N$) in the queue, then he can swap with the customer in position $i + D$, given that $i + D \leq N$.
3. He may repeat this swapping operation as many times as necessary.

The owner wishes to select the maximum possible distance D that still ensures he can successfully sort the entire line of customers by their ticket numbers.

Given the initial arrangement of the customers' ticket numbers from front to back, your task is to determine the maximum integer $D \leq N$ such that it is possible to completely sort the line using the Ice Cream Sort.

Input

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

The second line contains N distinct integers $T_1, T_2, \ldots, T_N$ ($1 \leq T_i \leq 10^9$), the ticket numbers of the customers in the line from front to back.

Output

Output a single integer, the maximum distance D ($1 \leq D \leq N$) such that the queue of customers can be sorted.

Examples

input	output
6 7 3 12 1 9 5	3
5 5 10 15 20 25	5

Note

In the first example, the sorted array is $[1, 3, 5, 7, 9, 12]$. If the owner chooses $D = 3$, he can sort the line using the following valid swaps between customers exactly 3 positions apart:

1. Swap 12 and 5: $[7, 3, 5, 1, 9, 12]$
2. Swap 7 and 1: $[1, 3, 5, 7, 9, 12]$

It can be proven that no larger D will allow the line to be completely sorted.

In the second example, the line is already sorted. The owner can choose $D = 5$. Since he doesn't need to make any swaps, any valid D works, and the maximum valid $D \leq N$ is 5.

J Jettisoned Joins

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
10TH · 25 JUL 2026

In databases, a table is a list of *rows*. Each row consists of the same set of *columns*, and within one table, no two columns share the same name.

For two database tables A and B (which may each have their own columns, some shared and some not), and a set of shared column names K called the *key columns*, the *inner join* I of A and B over K is defined as follows:

For each pair of rows (r_A, r_B) with r_A in A and r_B in B, if r_A and r_B have equal values in every column of K, then I contains one row combining all columns of r_A with all columns of r_B. Note that I is a list, i.e. if multiple pairs (r_A, r_B) satisfy the condition, each contributes a separate row to I, even if the resulting rows are identical.

Table 1

price	sku	store
100	3	9
250	3	9
70	4	9

Table 2

qty	sku	store
2	3	9
8	3	9

Inner Join

price	sku	store	qty
100	3	9	2
100	3	9	8
250	3	9	2
250	3	9	8

In the above example, there are two rows in each table with `sku` $= 3$ and `store` $= 9$. Hence, their inner join has $2 \times 2 = 4$ corresponding rows. There are no rows with `sku` $= 4$ and `store` $= 9$ in table 2, so there are no corresponding rows in their inner join.

Alice just learned what an inner join is in her database class! For her assignment, her teacher Bob gave her two database tables and asked her to compute their inner join. Alice computed it, but the result turned out to be too large to submit. Alice decides to delete **at most D rows in total**, taken from either table in any combination, before computing the join. Help Alice choose which rows to delete so that the resulting inner join has as few rows as possible, and report that minimum count.

Input

The first line consists of two integers R_1, C_1 ($1 \le R_1 \le 10^5, 1 \le C_1 \le 5$), the number of rows and columns of the first table.

The second line consists of C_1 distinct strings $N_{1,1}, \ldots, N_{1,C_1}$ ($1 \le |N_{1,i}| \le 10$), the column names of the first table.

Each of the next R_1 lines consists of C_1 integers $A_{i,1}, \ldots, A_{i,C_1}$ ($1 \le A_{i,j} \le 10^9$), where $A_{i,j}$ is the value in column $N_{1,j}$ of the i-th row of the first table.

The next line consists of two integers R_2, C_2 ($1 \le R_2 \le 10^5, 1 \le C_2 \le 5$), the number of rows and columns of the second table.

The next line consists of C_2 distinct strings $N_{2,1}, \ldots, N_{2,C_2}$ ($1 \le |N_{2,i}| \le 10$), the column names of the second table.

Each of the next R_2 lines consists of C_2 integers $B_{i,1}, \ldots, B_{i,C_2}$ ($1 \leq B_{i,j} \leq 10^9$), where $B_{i,j}$ is the value in column $N_{2,j}$ of the i-th row of the second table.

The next line consists of an integer K ($1 \leq K \leq \min(C_1, C_2)$), the number of key columns.

The next line consists of K distinct strings $s_1, \ldots, s_K$, the names of the key columns. It is guaranteed that $s_1, \ldots, s_K$ each appear as a column name in **both** tables.

The last line consists of an integer D ($0 \leq D \leq R_1 + R_2$), the maximum number of rows Alice may delete in total, across both tables.

All column names will only contain lowercase English letters.

Output

The output should consist of a single integer, the minimum number of rows in the resulting database table, after Alice deletes at most D rows optimally.

Examples

input

```
3 3
price sku store
100 3 9
250 3 9
70 4 9
2 3
qty sku store
2 3 9
8 3 9
2
sku store
0
```

output

```
4
```

input

```
4 2
region product
1 5
1 5
1 5
2 7
5 2
region product
1 5
2 7
2 7
2 7
2 7
2
region product
1
```

output

```
3
```

Note

For sample 1, we can refer to the example above. As we cannot delete any rows, the inner join must have a total of 4 rows.

For sample 2, we can delete the row with `region` $= 2$, `product` $= 7$ from table 1, and the resulting inner join will only have 3 rows. It can be proven that a better solution does not exist.

K KMP Evasion

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

The Knuth-Morris-Pratt algorithm (also known as KMP), in its simplest form, is used to solve the *Pattern-Matching Problem*: given a string S and pattern T, how many times does T occur as a substring of S?

The LSPCPC problemsetting team wants to include this problem in next year's contest, but sadly the judging servers cannot handle it when programs output big numbers. To help us, you are given a test case with string S (of length N) and pattern T (of length M). Please rearrange, or leave untouched, the characters in the string S, such that the answer to the *Pattern-Matching Problem* on the new shuffled string and pattern T is minimized. Output the shuffled string. If there are multiple solutions, you may output any of them.

Input

The first line contains two integers N and M ($1 \leq N, M \leq 2 \times 10^5$), representing the length of string S and pattern T respectively.

The second line contains the string S, which is to be shuffled.

The third line contains the pattern T.

All input strings are composed of lowercase characters only.

Output

Print a single line, containing any rearrangement of the string S that minimizes the number of times T occurs as a substring in it.

Examples

input

```
8 3
puilatiu
lsc
```

output

```
puitiula
```

input

```
5 2
aaaab
aa
```

output

```
aabaa
```

Note

In the first sample test, no matter how you arrange `puilatiu`, there is no `lsc` as a substring.

This page is intentionally left blank.

L Last Order

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

Mr. Noodle is the chef of a restaurant. In the restaurant, there are N types of food with different cooking times. The i-th type of food takes C_i seconds to cook.

The restaurant operates under the following rules:

- Mr. Noodle is the only person working in the restaurant and he is very busy, so at most one food can be ordered at each second.
- Mr. Noodle can receive orders while he is cooking. If an order is received at a second when Mr. Noodle is free, he can start cooking that new order immediately at that same second.
- Since Mr. Noodle is a hardworking person, he will continuously cook and never rest as long as there are unfinished orders.
- There is only one stove in the kitchen. It can cook multiple orders of the **same type** of food simultaneously, but it cannot be used to cook different types of food at the same time.
- When Mr. Noodle becomes free and there are multiple unfinished orders, he will cook the earliest placed order first.
- When Mr. Noodle starts cooking a food item of type X, he gathers all other unfinished orders of type X in the queue, and cooks them all in parallel on the stove.

Recently, Mr. Noodle found that he often needs to work overtime because there are still many unfinished orders after the restaurant closes. Therefore, Mr. Noodle decided to set a last order time T, which means that he will only accept orders on the first T seconds after the restaurant opens. Mr. Noodle wants to know if setting the last order time could prevent him from working overtime. As a friend of Mr. Noodle, please help him to determine the maximum possible completion time to finish all orders in the worst-case scenario, and provide a sequence of orders that achieves the maximum possible completion time.

Input

The first line contains two integers, N and T, representing the number of food types available and the last order time respectively. ($1 \leq N \leq 10^5$, $3 \leq T \leq 10^5$)

The second line contains N distinct integers, where the i-th integer denotes C_i, the cooking time in seconds for the i-th type of food. ($1 < C_i < T$)

Output

The output consists of two lines.

The first line contains a single integer representing the maximum possible completion time (in seconds).

The second line contains T integers, representing the optimal sequence of food types to order at each second. The i-th integer denotes the food type ordered at that second ($0 \leq O_i \leq N$), where 0 indicates that no food was ordered. If there are multiple solutions, output any of them.

Examples

input

```
2 6
2 3
```

output

```
12
1 1 2 2 1 2
```

input

```
3 8
5 2 4
```

output

```
22
2 3 1 0 0 2 3 1
```

Note

Explanation for Sample 1:

Time (t)	Queue	Cooking Status
1	[]	Food 1 (ordered at $t = 1$)
2	[1]	
3	[2]	Food 1 (ordered at $t = 2$)
4	[2, 2]	
5	[1]	Food 2 (ordered at $t = 3$ and $t = 4$)
6	[1, 2]	
7		
8	[2]	Food 1 (ordered at $t = 5$)
9		
10	[]	Food 2 (ordered at $t = 6$)
11		
12		