icpc International Collegiate Programming Contest



The 2026 ICPC Asia Seoul Regional Onsite Contest

Official Problem Set

JETBRAINS
icpc global sponsor
programming tools

HUAWEI
icpc diamond
world finals,
multi-regional sponsor

Jane Street
icpc multi-regional
titanium sponsor

Problem Set

Please check that you have 13 problems that span across 28 pages in total (including two cover pages).

A.	Baduk Solitaire	(2 pages)
B.	Boss Raid Log	(2 pages)
C.	Chemistry Experiment	(1 page)
D.	Circuit X	(2 pages)
E.	Drones	(2 pages)
F.	Erin Needs Cake	(4 pages)
G.	Glowing Devices	(2 pages)
H.	Last Jump	(2 pages)
I.	Potion Shop	(2 pages)
J.	Safe	(2 pages)
K.	Stepping Numbers	(1 page)
L.	Summoning Triangle	(2 pages)
M.	Workbench	(2 pages)

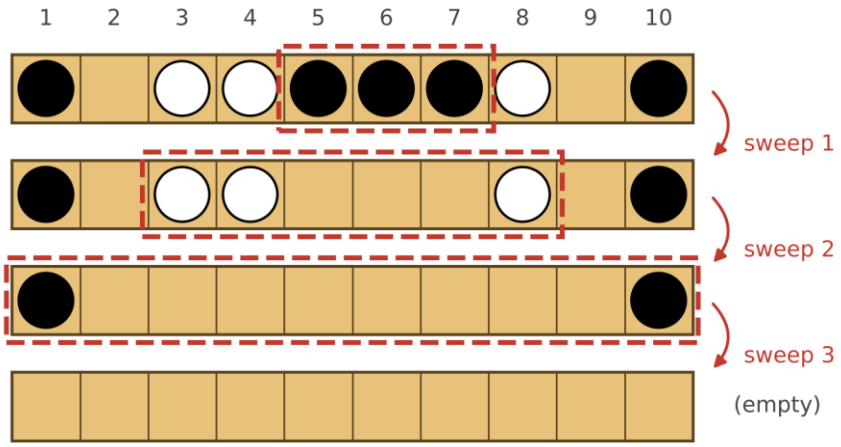
Problem A
Baduk Solitaire
Time Limit: 1 Second

Sumin found a long and narrow wooden board in her grandfather's attic. It is a single row cut from an old Baduk (Go) board and consists of n cells numbered 1 through n from left to right. Each cell is either empty or contains exactly one black or white Baduk stone. Sumin decides to play a solitaire puzzle on this board: she wants to remove all the stones from the board by repeatedly applying the only allowed move, called a *sweep*, which is defined as follows:

- Choose two distinct stones of the same color, say the stones in cells i and j with $i < j$.
- The sweep is allowed only if every stone in cells i through j, inclusive, has the same color as the two chosen stones. Empty cells between them do not matter.
- All the stones in cells i through j, inclusive, are removed from the board at once, and those cells become empty. The other stones stay in their cells.

Since a sweep always removes at least two stones of the same color, a stone can never be removed by itself; it must be swept together with another stone of the same color. Note that the order of sweeps matters.

For example, consider the board with $n = 10$ cells shown in the figure below. The board is represented by the string `B.WWBBBW.B`, where `B`, `W`, and `.` denote a black stone, a white stone, and an empty cell, respectively. Sumin can remove all the stones with three sweeps. She first sweeps the three black stones in cells 5 through 7. Then the white stones in cells 3, 4, and 8 can be swept together, so she sweeps them next. Finally, she sweeps the two black stones in cells 1 and 10. However, if her first sweep removes the two white stones in cells 3 and 4, or only two of the three black stones in cells 5 through 7, then she can never remove all the stones from the board.



Given the initial state of the board, write a program to determine whether Sumin can remove all the stones from the board.

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($1 \leq n \leq 1{,}000{,}000$), where n is the number of cells of the board.

The next line contains a string of length n that represents the board from cell 1 to cell n. Each character of the string is `B`, `W`, or `.`, which denotes a cell containing a black stone, a cell containing a white stone, or an empty cell, respectively. The board contains at least one stone.

Output

Your program is to write to standard output. Print exactly one line. If Sumin can remove all the stones from the board, print `YES`. Otherwise, print `NO`.

The following shows the sample input and output for four test cases.

Sample Input 1	Output for the Sample Input 1
1 B	NO

Sample Input 2	Output for the Sample Input 2
10 B.WWBBBW.B	YES

Sample Input 3	Output for the Sample Input 3
7 BB.WBWW	NO

Sample Input 4	Output for the Sample Input 4
15 BWB.WWBW.BBW.WB	YES

Problem B
Boss Raid Log
Time Limit: 1 Second

You are a wizard participating in a boss raid. There are k different types of spells, numbered from 1 to k. To defeat the boss, m spells must be cast in a specific order. Let $P = (P_1, P_2, \ldots, P_m)$ denote this required spell sequence, where each P_i denotes the type of the i-th spell in the sequence. The spells in the required sequence do not need to be cast by the same wizard, as different spells may be cast by different wizards. The same spell type may appear more than once in P.

During the raid, the wizards cast spells, and the raid log records their spell types in chronological order, but not the identities of the wizards who cast them. We assume that no two spells are cast at the same time. Suppose that spells are recorded at n different times during the battle. The resulting raid log is $S = (S_1, S_2, \ldots, S_n)$, where each S_i is a log entry representing the spell type recorded at the i-th time. For a successful attack, consecutive spells in the required sequence must be sufficiently separated in the raid log. Let d denote the minimum distance between the recorded times of two consecutive spells in the required sequence.

We say that the raid log contains a **successful attack** if there exist m recorded times $1 \le t_1 < t_2 < \cdots < t_m \le n$ satisfying both of the following conditions:

1. $S_{t_j} = P_j$ for every $1 \le j \le m$.
2. $t_{j+1} - t_j \ge d$ for every $1 \le j < m$.

A log entry S_i is covered if it is included in at least one successful attack; that is, if $i = t_j$ for some $1 \le j \le m$ in some successful attack. Finally, the raid log S is called consistent if every log entry of S is covered.

For example, suppose $d = 2$, $P = (1, 3, 2)$ and $S = (1, 1, 3, 3, 3, 2, 2, 2)$. Then the following are successful attacks: (a) recorded times $1, 3, 6$, (b) recorded times $2, 5, 7$, and (c) recorded times $2, 4, 8$. Every log entry of S is covered by at least one of these successful attacks, and hence S is consistent. On the other hand, suppose $d = 2$, $P = (1, 3, 2)$ and $S = (1, 3, 3, 3, 2, 2)$. The second log entry S_2 is not covered because it cannot be included in any successful attack. Therefore, S is not consistent.

Given k, d, the required spell sequence P, and the raid log S, write a program to determine whether S is consistent.

Input

Your program is to read from standard input. The input starts with a line containing four integers k, m, n, and d ($1 \le k \le 200{,}000$; $1 \le m \le n \le 200{,}000$; $1 \le d \le n$), which denote the number of spell types, the length of the required spell sequence, the number of entries in the raid log, and the minimum required distance between consecutive spells in a successful attack, respectively.

The next line contains m integers $P_1, P_2, \ldots, P_m$ ($1 \le P_j \le k$), which denote the required spell sequence.

The following line contains n integers $S_1, S_2, \ldots, S_n$ ($1 \le S_i \le k$), which denote the raid log.

Output

Your program is to write to standard output. Print exactly one line. Print `YES` if the raid log S is consistent and print `NO` otherwise.

The following shows the sample input and output for five test cases.

Sample Input 1

```
3 3 8 2
1 3 2
1 1 3 3 3 2 2 2
```

Output for the Sample Input 1

```
YES
```

Sample Input 2

```
3 3 6 2
1 3 2
1 3 3 3 2 2
```

Output for the Sample Input 2

```
NO
```

Sample Input 3

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

Output for the Sample Input 3

```
NO
```

Sample Input 4

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

Output for the Sample Input 4

```
YES
```

Sample Input 5

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

Output for the Sample Input 5

```
YES
```

Problem C
Chemistry Experiment
Time Limit: 3 Seconds

The International Chemical Protection Council (ICPC) enforces strict safety regulations in chemical laboratories: direct contact between different chemicals may trigger hazardous reactions, so a neutralizing agent must always be placed between them.

A laboratory has a row of n containment compartments, indexed from 1 to n. Each compartment must be filled with one of the following:

- One of k distinct chemical substances, labeled with integers from 1 to k.
- A neutralizing agent, labeled with the integer 0.

In this experiment, all k chemical substances must be utilized. Therefore, for every integer x from 1 to k, the chemical substance x **must be placed in at least one compartment**.

Let A_i denote the label of the chemical substance in compartment i, or 0 if the compartment contains the neutralizing agent. The resulting sequence is $A = (A_1, A_2, \ldots, A_n)$.

According to ICPC safety regulations, two adjacent compartments cannot contain different nonzero chemical substances. Formally, for every $1 \le i < n$, if $A_i \neq 0$ and $A_{i+1} \neq 0$, then $A_i = A_{i+1}$.

Given the integers n, k, and m, find the number of valid sequences A that satisfy all the conditions above, modulo m.

Input
Your program is to read from standard input. The first and only line of the input contains three space-separated integers n, k, and m ($1 \le k \le 5 \times 10^5$; $k \le n \le 10^9$; $10^8 < m < 10^9$; m is a prime number).

Output
Your program is to write to standard output. Print exactly one line. The line should contain the number of valid sequences modulo m.

The following shows the sample input and output for four test cases.

Sample Input 1	Output for the Sample Input 1
3 2 998244353	2

Sample Input 2	Output for the Sample Input 2
4 2 993244853	10

Sample Input 3	Output for the Sample Input 3
500000 123456 100000007	43861600

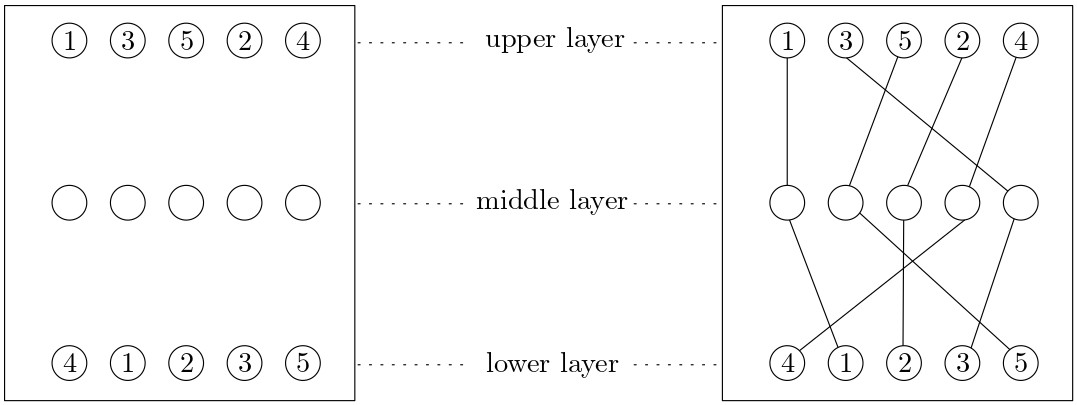
Sample Input 4	Output for the Sample Input 4
314159 265358 265531103	0

Problem D
Circuit X
Time Limit: 1 Second

You are leading a large circuit-design project. The project is nearly complete, and your final task is to finish its most important module, called Circuit X. Circuit X consists of three layers: an upper layer, a middle layer, and a lower layer. Each layer of Circuit X consists of n pins placed horizontally, from left to right. The pins in each of the upper and lower layers are labeled with a permutation of the integers from 1 to n. The pins in the middle layer are unlabeled.

To complete Circuit X, for each $1 \le i \le n$, connect the upper-layer pin labeled i to a middle-layer pin with a straight wire, and connect that same middle-layer pin to the lower-layer pin labeled i with another straight wire. Thus, exactly $2n$ wires are used. Do not forget that each pin in the middle layer must be connected to one and only one pin in the upper layer and to one and only one in the lower layer.

The figure below shows an example of Circuit X in which $n = 5$. On the left, the current configuration of Circuit X is shown, where the upper-layer pins are labeled $(1, 3, 5, 2, 4)$ from left to right, and the lower-layer pins are labeled $(4, 1, 2, 3, 5)$. The right part shows a possible completion of Circuit X.



There are many possible ways to complete Circuit X, depending on how the labeled pins are matched to the middle-layer pins. You need to be very careful because your project will be evaluated by the *complexity* of the final form of Circuit X.

The complexity of Circuit X is determined by the number of crossings among wires. Specifically, for a fixed completion of Circuit X, let $\chi^{\wedge}$ be the number of pairs of crossing wires between the upper and middle layers, and $\chi^{\vee}$ be the number of pairs of crossing wires between the middle and lower layers. Then the complexity of Circuit X is defined to be $a\chi^{\wedge} + b\chi^{\vee}$, where a and b are two predefined integer parameters. In the above example, we have $\chi^{\wedge} = 3$ and $\chi^{\vee} = 5$, so the complexity of Circuit X will be $3a + 5b$ depending on a and b. Note that the complexity is not necessarily positive, since a and b may be zero or negative.

To obtain the best evaluation for your project, you need to minimize the complexity of Circuit X. Given the orders of the labels of pins in the upper and lower layers of Circuit X and two integer parameters a and b, write a program to output the minimum possible complexity of Circuit X.

Input

Your program is to read from standard input. The input consists of three lines. The first line consists of three integers, n, a, and b ($2 \leq n \leq 1{,}000{,}000$; $-10{,}000 \leq a, b \leq 10{,}000$), where n is the number of pins in each layer of Circuit X, and a, b are integer parameters for the complexity of Circuit X.

The second line consists of a permutation of integers $\{1, 2, \ldots, n\}$, which represents the labels of the n pins in the upper layer of Circuit X from left to right.

The third line consists of another permutation of integers $\{1, 2, \ldots, n\}$, which represents the labels of the n pins in the lower layer of Circuit X from left to right.

Output

Your program is to write to standard output. Print exactly one line consisting of a single integer that represents the minimum possible complexity of the input instance of Circuit X.

The following shows the sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
5 3 4 1 3 5 2 4 4 1 2 3 5	18

Sample Input 2	Output for the Sample Input 2
4 -1 0 1 2 3 4 4 3 2 1	-6

Problem E
Drones
Time Limit: 0.5 Seconds

The military base is represented by a simple polygon with at most 5 sides. Note that a simple polygon is a polygon whose boundary does not intersect itself. More precisely, its edges intersect only at their common endpoints when they are consecutive edges, and nonconsecutive edges do not intersect. No three consecutive vertices of the polygon are collinear.

The base monitors whether nearby drones are approaching its boundary. The plane is divided into three disjoint zones: the "Inner" zone, consisting of the interior and boundary of the base polygon; the "Boundary" zone, consisting of points strictly outside the base polygon whose shortest distance to its boundary is at most d; and the "Outer" zone, consisting of all remaining points. In particular, points on the polygon boundary belong to the Inner zone, while points outside the polygon at distance exactly d from its boundary belong to the Boundary zone. The distance between a drone and the base is defined as the shortest distance from the drone to the base's boundary. For every point in the Outer zone, its shortest distance l to the polygon boundary satisfies $l > d$.

All polygon vertices and drones have integer coordinates. The polygon vertices are pairwise distinct, and the drone positions are also pairwise distinct. A drone may have the same coordinates as a polygon vertex. Figure 1 illustrates two example bases. The boundary depicted by the dotted line outside the base represents the limit of the "Boundary" zone.

For Base A, the drones located in the Inner, Boundary, and Outer zones are $\{D_3, D_6, D_8\}$, $\{D_1, D_2, D_7\}$, and $\{D_4, D_5\}$, respectively; for Base B, they are $\{D_1, D_2\}$, $\{D_5, D_6, D_7\}$, and $\{D_3, D_4\}$. The polygon boundary belongs exclusively to the Inner zone, and the outer boundary of the Boundary zone belongs exclusively to the Boundary zone.

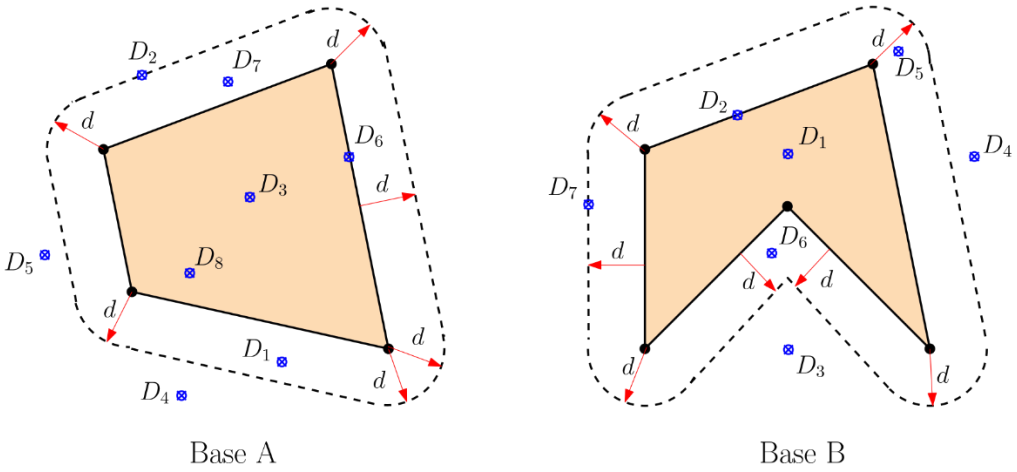


Figure 1. Two example bases with drones (blue dots).

Given the coordinates of base vertices in counterclockwise order and the coordinates of n observed drones, write a program to determine the number of drones located in each of the three zones: Inner, Boundary, and Outer.

Input

Your program is to read from standard input. The first line contains two integers s and d ($3 \leq s \leq 5$; $1 \leq d \leq 10{,}000$), where s is the number of sides of the base polygon and d is the maximum distance defining the Boundary zone.

The second line contains $2s$ integers $x_1, y_1, x_2, y_2, \ldots, x_s, y_s$ ($1 \leq x_i, y_i \leq 10{,}000$), representing the coordinates (x_i, y_i) of the polygon's vertices in counterclockwise order.

The third line contains an integer n ($1 \leq n \leq 1{,}000$), the number of drones.

Each of the next n lines contains two integers x_j' and y_j' ($1 \leq x_j', y_j' \leq 10{,}000$), the coordinates (x_j', y_j') of a drone.

Output

Your program is to write to standard output. Print exactly one line. Print three space-separated integers: the numbers of drones in the Inner, Boundary, and Outer zones, respectively.

The following shows the sample input and output for two test cases corresponding to Bases A and B, respectively.

Sample Input 1

```
4 65
260 300 800 180 680 800 200 600
8
580 180
415 760
500 500
400 100
140 380
740 490
470 730
370 350
```

Output for the Sample Input 1

```
3 3 2
```

Sample Input 2

```
5 65
200 200 500 500 800 200 680 800 200 620
7
500 610
440 710
500 200
900 600
715 825
500 430
135 500
```

Output for the Sample Input 2

```
2 3 2
```

Problem F
Erin Needs Cake
Time Limit: 3 Seconds

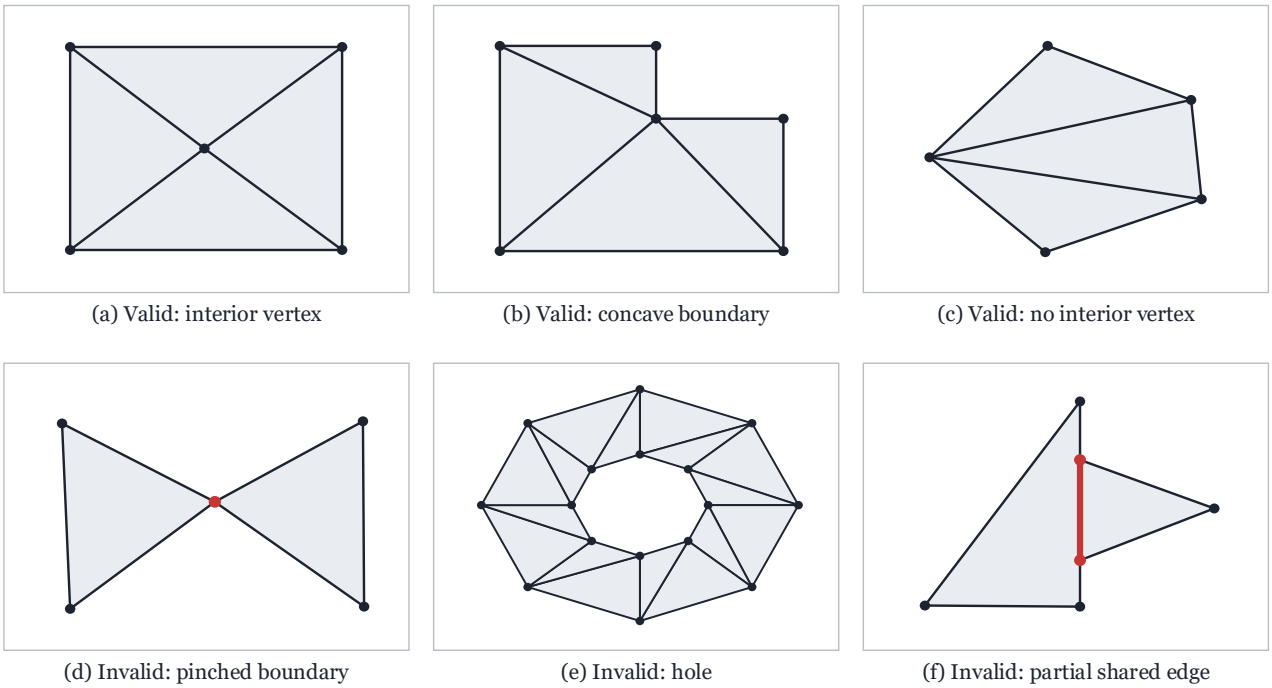


Erin, the queen of the fairies in *Elias*, had a cake she was saving for later. To keep anyone else from eating it, she hid it in a secret place. Unfortunately, she hid it so well that she has now forgotten where it is.

Fortunately, Erin left a locator charm with her map. The map is divided into triangular regions, and the cake is hidden strictly inside exactly one of them, never on a road or a vertex. Trace a closed loop along the map's roads, and the charm will tell you whether the cake is inside the loop.

Each triangle has positive area, and distinct triangles have disjoint interiors. Together, the triangles form a single simple polygon without holes. In particular, the outer boundary is a single simple cycle, so no vertex appears more than once along the boundary. Every vertex belongs to at least one triangle. Each triangle side is a road; a side shared by two triangles is a single road. Distinct roads meet only at common endpoints. The outer boundary need **not** be convex, and vertices may lie strictly inside it.

Figure 1 shows examples of valid and invalid maps.



(a) Valid: interior vertex
(b) Valid: concave boundary
(c) Valid: no interior vertex
(d) Invalid: pinched boundary
(e) Invalid: hole
(f) Invalid: partial shared edge

Figure 1. Examples of valid and invalid maps.

Help Erin identify the triangular region containing her cake using at most $\lceil \log_2 n \rceil$ queries, where n is the number of regions. Here, $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

Interaction

This is an interactive problem. Your submitted program will interact with an interactor inside the grading server, which reads input from and writes output to your program. Your program must gather information by printing queries and reading the interactor's replies. After every line of output, your program must immediately flush the output.

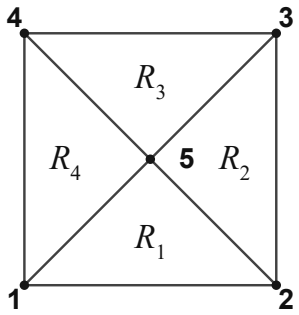
Initial Input

The first line contains two integers n and k: the number of triangular regions and the number of vertices, respectively ($2 \le n \le 2^{18}$; $4 \le k \le n + 2$).

Vertices are numbered from 1 to k, and triangular regions are numbered from 1 to n.

The next k lines describe the vertices. For each i ($1 \le i \le k$), the i-th of these lines contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$), the coordinates of vertex i. All vertices have distinct coordinates.

The next n lines describe the triangular regions. For each i ($1 \le i \le n$), the i-th of these lines contains three distinct integers u_i, v_i, and w_i ($1 \le u_i, v_i, w_i \le k$), the vertices of region i.



Black dots are vertices. Bold numbers label vertices; R_1–R_4 label triangular regions.

Region	Vertices in the input
R_1	$1, 2, 5$
R_2	$2, 3, 5$
R_3	$3, 4, 5$
R_4	$4, 1, 5$

Here $n = 4$ and $k = 5$. Vertex 5 is at $(2, 2)$, inside the map. The area outside the square is not a region.

Figure 2. The map in Sample Interaction 1, drawn from its input coordinates.

Queries

Your program may repeatedly output exactly one of the following commands until the interactor outputs "`correct`" or "`wrong`". Every number and special character ("`?`", "`!`") used in the query should be separated by a single space.

- "`?` l p_1 p_2 ... p_l"
  - Ask whether the cake is inside the polygon whose boundary follows vertices $p_1, p_2, \ldots, p_l$ in that order and then returns to p_1.
  - Here $3 \le l \le k$, and $p_1, p_2, \ldots, p_l$ must be **distinct** vertex indices. Every consecutive pair, including $\{p_l, p_1\}$ must be joined by a road. The boundary must be one simple loop: it cannot cross itself or visit a vertex twice. Either direction is allowed. Do **not** repeat p_1 at the end; the closing road is implicit. A straight path through another vertex is **not** a single road.
- "`!` a"
  - Report the region number a containing the cake ($1 \le a \le n$).
  - Flush this command, read the interactor's final reply, and terminate.

Each command must be printed on its own line and immediately flushed. You may ask at most $\lceil \log_2 n \rceil$ queries starting with "`?`". The final "`!`" command does not count toward this limit and must be issued exactly once, unless you have already received "`wrong`".

Interactor Responses

After each command, the interactor responds as follows:

- For a valid "`?`" query, the interactor outputs "`1`" if the cake is inside the chosen polygon, and "`0`" otherwise.
- For a query "`! a`", the interactor outputs "`correct`" if you have successfully identified the hidden region, and "`wrong`" otherwise.
- The interactor immediately terminates the interaction after printing "`wrong`" if any rule is violated — for example, a malformed command, an out-of-range index, or exceeding $\lceil \log_2 n \rceil$ queries starting with "`?`".

The interactor **may be adaptive**. It need not choose a hiding place in advance, but all replies together must be consistent with at least one fixed region. The cake does not move. Your answer must be forced by the replies: guessing one of several still-possible regions is not sufficient.

The time and memory used by the interactor are also included in the calculation of your program's execution time and memory usage. You can assume that the maximum time used by the interactor is 1 second and the maximum amount of memory is 64 MiB.

The following interaction uses the map in Figure 2.

Sample Interaction 1

Read (Interactor's Response)	**Write (Your Command)**

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

	`? 4 1 2 3 5`
`0`	
	`? 3 3 4 5`
`1`	
	`! 3`
`correct`	

To flush, your program needs to do the following right after writing a command and a newline:

- `fflush(stdout)` in C;
- `std::cout << std::flush` in C++;
- `System.out.flush()` in Java or Kotlin;
- `sys.stdout.flush()` in Python.

A testing tool is provided to help you develop your solution, so it is not mandatory to use it. If you want to use it, you can download the attachment `testing_tool.py` from the *DOMjudge Problemset* page. You can run the testing tool to see how your program interacts with the interactor on a specific map and a fixed hiding region for the cake. **This testing tool can be used regardless of the language of your program as follows.** The same explanation is also included in the comments of `testing_tool.py`.

Usage: `python3 testing_tool.py -f <inputfile> -- <program>`

Use the `-f` parameter to specify the input file, e.g. `input.in`.

Format of the input file: The first line contains two integers n and k. For each i ($1 \leq i \leq k$), the i-th of the next k lines contains two integers x_i and y_i. For each i ($1 \leq i \leq n$), the i-th of the next n lines contains three integers u_i, v_i, and w_i. The last line contains an integer h ($1 \leq h \leq n$), indicating the index of the region where the cake is hidden.

The final integer h is used only by the testing tool and is not provided to your program.

Example: The map from the sample interaction is represented as follows.

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

The tool is provided as-is, and you should feel free to make whatever alterations or augmentations you like to it. **Note that it is not guaranteed that a program that passes the testing tool will be accepted**. For example, the testing tool uses a **fixed hiding region** specified in the input file, whereas the grading interactor may respond adaptively.

If you have a C++ solution stored in a file called "`sol.cpp`", you must first compile using "`g++ sol.cpp -o sol`" and then invoke the testing tool with:

```
python3 testing_tool.py -f input.in -- ./sol
```

If you have a Python solution that you would run using "`pypy3 solution.py`", you can invoke the testing tool with:

```
python3 testing_tool.py -f input.in -- pypy3 solution.py
```

If you have a Java solution that you would run using "`java MyClass`", you can invoke the testing tool with:

```
python3 testing_tool.py -f input.in -- java MyClass
```

If you have a Kotlin solution that you would run using "`kotlin SolutionKt`", you can invoke the testing tool with:

```
python3 testing_tool.py -f input.in -- kotlin SolutionKt
```

Problem G
Glowing Devices
Time Limit: 1 Second

A lab has n self-luminous devices, numbered from 1 to n. Each device is either **glowing** or **dark**. Once a device starts glowing, it never goes dark again.

A dark device does not need an external power source to start glowing. It is triggered by the light of the other devices. Device i has a **threshold** t_i ($1 \le t_i \le n-1$). As soon as at least t_i devices are glowing, device i starts glowing too.

The devices are initially dark, and none of them can start the (glowing) chain on its own. To start the chain, you pick a set S of devices and **force** each of them to glow. Forcing device i costs w_i. The devices in S start glowing all at once, and every other device starts out dark. After that, the following rule is applied over and over:

> Let L be the number of devices that are currently glowing (including the devices in S). If some dark device i satisfies $L \ge t_i$, device i starts glowing.

The process stops when no dark device satisfies this condition. A glowing device never goes dark, so L never decreases. Therefore, the set of devices glowing at the end depends only on S, not on the order in which devices start glowing.

A set S is **valid** if all n devices are glowing when the process stops. The cost of S is $\sum_{i \in S} w_i$.

Given thresholds and costs of devices, find the minimum cost of a valid set S.

Forcing every device, $S = \{1, 2, \ldots, n\}$, is always valid, so an answer always exists. Since every threshold is at least 1, no device starts glowing unless at least one device is forced, so S must be non-empty.

Input
Your program is to read from standard input. The input starts with a line containing an integer n ($2 \le n \le 100{,}000$), where n is the number of devices. The devices are numbered from 1 to n.

The second line contains n integers $t_1, t_2, \ldots, t_n$ ($1 \le t_i \le n-1$), the thresholds of the devices.

The third line contains n integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le 10^9$), the costs of forcing each device to glow.

Output
Your program is to write to standard output. Print exactly one line. The line should contain one integer that is the minimum total cost of the devices you force to glow so that every device ends up glowing.

The following shows the sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
3 1 1 1 4 2 7	2

Sample Input 2	Output for the Sample Input 2
3 1 1 1 3 3 3	3

Sample Input 3	Output for the Sample Input 3
6 3 4 3 1 4 3 3 2 8 1 4 5	5

Problem H
Last Jump
Time Limit: 1 Second

Two players Alice and Bob play the following game with a length-n string $T = T[1] \circ T[2] \circ \cdots \circ T[n]$, where $T[i]$ belongs to the alphabet Σ that consists of 26 lowercase Latin letters, and $\circ$ is the concatenation operator.

1. The game starts with an empty string X and an integer $i = 0$, and Alice makes the first move.
2. The players alternate turns. On each turn, the current player chooses a character $c \in \Sigma$ and proceeds as follows.
   ① If $T[i+1] = c$, then increment i by 1.
   ② Otherwise, the player attempts a ***jump*** as follows.
      i. Find the minimum j such that
         (a) $j > i$, and
         (b) $T[j-m] \circ T[j-m+1] \circ \cdots \circ T[j] = X \circ c$, where m is the length of X.
      ii. If no such j exists, the game ends immediately and the current player loses.
      iii. Update i to j.
   ③ In both cases, update X to $X \circ c$.
3. If a successful move makes $i = n$, the game ends. The player who made the most recent jump wins. If no jump has occurred, Alice wins. Every successful move strictly increases i, so the game ends after at most n successful moves.

The following table illustrates an example game:

Turn	State	Note
Start	$i = 0$ $T =$ a b a a b a b a a a b	
1 (Alice)	$i = 2$ $T =$ a b a a b a b a a a b $X =$ b	Alice chooses b (jump)
2 (Bob)	$i = 3$ $T =$ a b a a b a b a a a b $X =$ b a	Bob chooses a
3 (Alice)	$i = 4$ $T =$ a b a a b a b a a a b $X =$ b a a	Alice chooses a
4 (Bob)	$i = 10$ $T =$ a b a a b a b a a a b $X =$ b a a a	Bob chooses a (jump)
5 (Alice)	$i = 11$ $T =$ a b a a b a b a a a b $X =$ b a a a b	Alice chooses b (Game ends, Bob wins)

Bob wins this example game because he made the last jump, on Turn 4. However, note that if Alice had chosen `b` instead of `a` on Turn 3, Alice would have won. In fact, if both players play their best, the winner of the game with string `abaababaaab` is Alice.

Given k strings, write a program to output the winner of the game for each of the k strings when both players play their best.

Input

Your program is to read from standard input. The input starts with a line containing one integer k ($1 \leq k \leq 10$) that indicates the number of strings.

Each of the next k lines contains a string of n lowercase Latin letters ($1 \leq n \leq 100{,}000$). The lengths of the strings may differ.

Output

Your program is to write to standard output. Print exactly one line for each string. Each line should contain exactly one capital Latin letter, either `A` if the winner is Alice or `B` if the winner is Bob, when the two players play their best with the corresponding string.

The following shows the sample input and output for three test cases.

Sample Input 1

```
4
aaaaa
ababb
baabb
bbaab
```

Output for the Sample Input 1

```
A
A
B
B
```

Sample Input 2

```
4
abaababaaab
aabcaabbcca
abcaacbbbca
aabbaaccaac
```

Output for the Sample Input 2

```
A
B
A
B
```

Sample Input 3

```
4
aaaaaaaabbbbb
xxxxyyyyxxxxyy
zzhhzzhhzzhhzzh
ststststststststst
```

Output for the Sample Input 3

```
A
B
B
A
```

Problem I
Potion Shop
Time Limit: 1 Second

Two apprentices at a potion shop play a game using the remaining ingredients. There are n containers, and the i-th container holds a_i magic crystals.

The players alternate turns. On each turn, the current player must make exactly one potion using either of the following methods:

1. **Specialized potion:** Choose a nonempty container and remove any positive number of crystals from it, up to the number currently in that container.
2. **Lesser potion:** Remove a total of between 1 and $p-1$ crystals, inclusive, from any number of containers. The crystals may be taken from a single container or distributed among multiple containers.

A player cannot use more crystals from a container than it currently holds. Used crystals are removed. A player who cannot make a potion on their turn loses. Both players play optimally.

For example, suppose there are two containers, each holding one crystal, and $p = 3$. The first player can remove both crystals in a single lesser-potion move, leaving no legal move for the second player. This corresponds to Sample Input/Output 1.

After the initial inventory, q updates are performed. Each update sets the number of crystals in one container to a given value. The new value may be smaller than, equal to, or greater than the current value. Updates are applied cumulatively in the order given in the input. Determine the winner of a game starting from the initial inventory and from the inventory immediately after each update. These hypothetical games do not change the inventory.

Input
Your program is to read from standard input. The first line contains three integers n, p, and q ($1 \le n \le 200{,}000$; $2 \le p \le 10^9$; $0 \le q \le 200{,}000$), where n is the number of containers, p is the exclusive upper bound on the number of crystals used in a lesser potion, and q is the number of updates.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^{18}$), the initial numbers of crystals in the containers.

Each of the following q lines contains two integers x and v ($1 \le x \le n$; $0 \le v \le 10^{18}$). This update sets the number of crystals in container x to v.

Output
Your program is to write to standard output. Print $q+1$ lines, one for the initial inventory and one for the inventory after each update, in order. Print `FIRST` if the first player has a winning strategy; otherwise, print `SECOND`.

The following shows the sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
2 3 0 1 1	FIRST

Sample Input 2	Output for the Sample Input 2
4 2 1 0 1 0 1 1 5	SECOND FIRST

Sample Input 3	Output for the Sample Input 3
2 3 4 0 0 1 1 2 2 1 0 1 7	SECOND FIRST SECOND FIRST FIRST

Problem J
Safe
Time Limit: 1 Second

An intelligence agency has obtained a locked safe containing secret documents. Attached to the safe is a sheet of paper containing two sequences of digits, X and Y, and k tables of operation costs. An intelligence analyst has discovered that the code to unlock the safe can be determined as follows.

Both X and Y consist of digits from 1 to 9, inclusive. Each cost table specifies the costs of three operations on a sequence of digits:

- **Replacement:** Replace one digit with another digit.
- **Insertion:** Insert a digit at any position.
- **Deletion:** Delete a digit from any position.

Each operation has a nonnegative integer cost. For a fixed cost table, let $\delta(A, B)$ denote the minimum total cost of transforming sequence A into sequence B using any number of these operations, including zero operations.

Each cost table is represented by a 10×10 matrix C. Its rows and columns are indexed, in order, by $1, 2, \ldots, 9, 0$, where 0 represents the empty symbol. The entries are interpreted as follows:

- For $1 \le a, b \le 9$, $C_{a,b}$ is the cost of replacing digit a with digit b.
- For $1 \le a \le 9$, $C_{a,0}$ is the cost of deleting digit a.
- For $1 \le b \le 9$, $C_{0,b}$ is the cost of inserting digit b.

The costs satisfy the following properties:

1. $C_{a,a} = 0$ for every $a \in \{1, 2, \ldots, 9, 0\}$.
2. $C_{a,b} = C_{b,a}$ for every $a, b \in \{1, 2, \ldots, 9, 0\}$.
3. $C_{a,c} \le C_{a,b} + C_{b,c}$ for every $a, b, c \in \{1, 2, \ldots, 9, 0\}$.

Now, partition Y into $p \ge 2$ nonempty contiguous subsequences $Y_1, Y_2, \ldots, Y_p$, such that $Y = Y_1 Y_2 \ldots Y_p$. For a fixed cost table C and a fixed partition, define the partition cost as $\max_{1 \le i \le p} \delta(X, Y_i)$. The candidate code for a cost table is the minimum partition cost over all possible choices of p and all valid partitions of Y.

Finally, the code required to unlock the safe is the minimum candidate code among the k cost tables.

For example, suppose $X = 12345$, $Y = 123412345642345$, and the three cost tables are as shown in the figure below.

X=12345, Y=123412345642345

	1	2	3	4	5	6	7	8	9	0
1	0	1	2	3	4	5	6	7	8	9
2	1	0	1	2	3	4	5	6	7	8
3	2	1	0	1	2	3	4	5	6	7
4	3	2	1	0	1	2	3	4	5	6
5	4	3	2	1	0	1	2	3	4	5
6	5	4	3	2	1	0	1	2	3	4
7	6	5	4	3	2	1	0	1	2	3
8	7	6	5	4	3	2	1	0	1	2
9	8	7	6	5	4	3	2	1	0	1
0	9	8	7	6	5	4	3	2	1	0

	1	2	3	4	5	6	7	8	9	0
1	0	4	4	4	4	4	4	4	4	4
2	4	0	4	4	4	4	4	4	4	4
3	4	4	0	4	4	4	4	4	4	4
4	4	4	4	0	4	4	4	4	4	4
5	4	4	4	4	0	4	4	4	4	4
6	4	4	4	4	4	0	4	4	4	4
7	4	4	4	4	4	4	0	4	4	4
8	4	4	4	4	4	4	4	0	4	4
9	4	4	4	4	4	4	4	4	0	4
0	4	4	4	4	4	4	4	4	4	0

	1	2	3	4	5	6	7	8	9	0
1	0	1	2	3	4	5	5	5	5	5
2	1	0	1	2	3	4	5	5	5	5
3	2	1	0	1	2	3	4	5	5	5
4	3	2	1	0	1	2	3	4	5	5
5	4	3	2	1	0	1	2	3	4	5
6	5	4	3	2	1	0	1	2	3	4
7	5	5	4	3	2	1	0	1	2	3
8	5	5	5	4	3	2	1	0	1	2
9	5	5	5	5	4	3	2	1	0	1
0	5	5	5	5	5	4	3	2	1	0

For the first table, partitioning Y into 12341, 23456, and 42345 gives transformation costs 4, 5, and 3, respectively. The minimum possible partition cost for this table is 5.

For the second table, partitioning Y into 1234, 123456, and 42345 gives transformation costs 4, 4, and 4, respectively. The minimum possible partition cost for this table is 4.

For the third table, the same partition gives transformation costs 5, 4, and 3, respectively. The minimum possible partition cost for this table is 5.

Therefore, the code to unlock the safe is $\min\{5, 4, 5\} = 4$.

Given the two sequences X and Y and the k cost tables, write a program to determine the code to unlock the safe.

Input

Your program is to read from standard input. The first line contains two strings X and Y, followed by an integer k ($3 \leq |X| \leq 10$; $2|X| \leq |Y| \leq 10{,}000$; $2 \leq k \leq 100$). Both strings consist only of digits from 1 to 9, inclusive.

The next $10k$ lines describe the k cost tables, with 10 consecutive lines for each table. For each table, the i-th line contains 10 space-separated integers, representing the operation costs in the row corresponding to digit i for $1 \leq i \leq 9$, or to the empty symbol for $i = 10$. The columns correspond, in order, to digits $1, 2, \ldots, 9$, followed by the empty symbol. Each cost is an integer between 0 and 9, inclusive.

It is guaranteed that every cost table satisfies the properties stated above.

Output

Your program is to write to standard output. Print exactly one integer on a single line: the code required to unlock the safe.

The following shows partial sample inputs and their corresponding outputs for three test cases. The complete sample input and output files are available on the *DOMjudge Problemset* page.

Sample Input 1

```
12345 123412345642345 3
0 1 2 3 4 5 6 7 8 9
1 0 1 2 3 4 5 6 7 8
... (28 more lines)
```

Output for the Sample Input 1

```
4
```

Sample Input 2

```
598 499281 2
0 4 5 2 3 7 5 3 5 5
4 0 4 4 3 7 3 5 4 1
... (18 more lines)
```

Output for the Sample Input 2

```
8
```

Sample Input 3

```
5758 435198918 2
0 1 3 2 2 2 5 2 3 1
1 0 2 1 1 3 5 3 4 2
... (18 more lines)
```

Output for the Sample Input 3

```
8
```

Problem K
Stepping Numbers
Time Limit: 1 Second

A positive integer is called a stepping number if, in its decimal representation without leading zeros, the absolute difference between every pair of adjacent digits is exactly 1. For example, both an increasing step (a 2 following a 1) and a decreasing step (a 1 following a 2) are allowed. A single-digit number (from 1 to 9) has no adjacent pair to compare, so it is also considered a stepping number.

For example, 10, 21, 123432, and 56565 are stepping numbers. On the other hand, 11 (difference 0), 13 (difference 2), and 100 (its last two digits 0 and 0 have difference 0) are not stepping numbers.

Given an integer n, you must find the n-th smallest stepping number.

Input
Your program is to read from standard input. The input consists of a single line containing one integer n ($1 \leq n \leq 10^5$).

Output
Your program is to write to standard output. Print the n-th smallest stepping number on a single line.

It is guaranteed that the answer is less than 10^{15}, so it fits in a signed 64-bit integer.

The following shows the sample input and output for five test cases.

Sample Input 1	Output for the Sample Input 1
1	1

Sample Input 2	Output for the Sample Input 2
10	10

Sample Input 3	Output for the Sample Input 3
12	21

Sample Input 4	Output for the Sample Input 4
100	6765

Sample Input 5	Output for the Sample Input 5
100000	321210121012343

Problem L
Summoning Triangle
Time Limit: 1 Second

In the kingdom of ICPC, n magic stones are scattered across the land. Each magic stone is located at a point on a plane and possesses exactly one of the k different elements. No three magic stones are collinear.

To defeat the evil "Demon Lord," the kingdom must summon heroes from another world using summoning triangles. A summoning triangle is a triangle whose three vertices are magic stones. For a successful summoning, the triangle must contain exactly one magic stone of each of the k elements, counting the stones at its vertices and in its interior. Since no three magic stones are collinear, no other stone can lie on an edge of the triangle.

Given the positions and elements of n magic stones, count the number of summoning triangles that can successfully summon a hero.

Input
Your program is to read from standard input. The input starts with a line containing two integers, n and k ($3 \le n \le 300$; $3 \le k \le n$), where n is the number of magic stones and k is the number of elements.

Each of the next n lines contains three integers x, y ($0 \le x, y \le 50{,}000$), and z ($1 \le z \le k$) that represent the coordinates (x, y) and element z of a magic stone. It is guaranteed that all magic stones are located at distinct points, no three magic stones are collinear, and at least one magic stone of each element is present.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the number of summoning triangles that can successfully summon a hero.

The following shows the sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
3 3 1 1 1 0 0 2 2 1 3	1

Sample Input 2	Output for the Sample Input 2
4 4 0 0 1 1 1 2 0 1 3 1 0 4	0

Sample Input 3	Output for the Sample Input 3
5 4 3 3 4 0 3 2 6 2 3 5 7 2 5 5 1	2

Problem M
Workbench
Time Limit: 1 Second

A workshop has a storeroom and a workbench. It turns a raw part into a finished product through n processing steps, numbered from 1 to n, which are carried out in this order. There are $n + 1$ parts, numbered from 0 to n: step i takes part $i - 1$ and produces part i, so part 0 is the raw material and part n is the finished product. Step i involves f_i units of work, and part i has size s_i.

The steps are carried out in *batches*. That is, the steps $1, \ldots, n$ are partitioned into intervals, and the batches are carried out one after another in the order of the intervals. A batch $[l, r]$ consists of steps l through r, inclusive. Its input part $l - 1$ is brought from the storeroom to the workbench, and its output part r is put back into the storeroom. Its intermediate parts $l, \ldots, r - 1$ never leave the workbench.

Let m be the area of the workbench. While a batch $[l, r]$ is carried out, all of its intermediate parts must be on the workbench at the same time, so the batch is *valid* only if $s_l + \cdots + s_{r-1} \le m$. The input and output parts do not count toward this area. A batch of a single step has no intermediate parts, so it is always valid.

The workshop has a *processing rate* a, the amount of work done per unit time, and a *transport rate* b, the total size of parts carried between the storeroom and the workbench per unit time. For simplicity, we assume that processing and transport within a batch fully overlap. Thus, the time taken by a batch $[l, r]$ is the larger of the processing time $(f_l + \cdots + f_r)/a$ and the transport time $(s_{l-1} + s_r)/b$. Multiplying this time by $a \times b$ gives an integer, which we call the *cost* of the batch:

$$\text{cost}(l, r) = \max\big((f_l + \cdots + f_r) \times b,\ (s_{l-1} + s_r) \times a\big)$$

For example, suppose that $a = 5$ and $b = 8$, and consider a batch of two steps involving 11 and 10 units of work. If its input and output parts have sizes 8 and 11, respectively, the cost of the batch is $\max(21 \times 8, 19 \times 5) = 168$.

The total cost of a partition is the sum of the costs of its batches. Since carrying out every step as a separate batch is always valid, at least one partition of the steps into valid batches exists. Write a program to find the minimum total cost over all partitions of the steps into valid batches.

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($1 \le n \le 2 \times 10^5$), where n is the number of steps.

The second line contains three integers a, b, and m ($1 \le a, b \le 10^6$; $0 \le m \le 10^{12}$), where a is the processing rate, b is the transport rate, and m is the area of the workbench.

The third line contains n integers $f_1, f_2, \ldots, f_n$ ($1 \le f_i \le 10^6$ for $1 \le i \le n$), where f_i is the amount of work involved in step i.

The fourth line contains $n + 1$ integers $s_0, s_1, \ldots, s_n$ ($1 \le s_i \le 10^6$ for $0 \le i \le n$), where s_i is the size of part i.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum total cost over all partitions of the steps into valid batches.

The following shows the sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
3 5 8 15 11 10 5 8 12 11 12	220

Sample Input 2	Output for the Sample Input 2
4 5 5 2 1 2 1 3 2 1 2 1 3	45