Problem A
Decimal Sequences

Input: Standard Input
Time Limit: 1 second

Hanako learned the conjecture that all the non-negative integers appear in the infinite digit sequence of the decimal representation of $\pi = 3.14159265\cdots$, the ratio of a circle's circumference to its diameter. After that, whenever she watches a sequence of digits, she tries to count up non-negative integers whose decimal representations appear as its subsequences.

For example, given a sequence "`3 0 1`", she finds representations of five non-negative integers 3, 0, 1, 30 and 301 that appear as its subsequences.

Your job is to write a program that, given a finite sequence of digits, outputs the smallest non-negative integer not appearing in the sequence. In the above example, 0 and 1 appear, but 2 does not. So, 2 should be the answer.

Input

The input consists of a single test case.

n
$d_1\ d_2\ \cdots\ d_n$

n is a positive integer that indicates the number of digits. Each of d_k's $(k = 1, \ldots, n)$ is a digit. There is a space or a newline between d_k and d_{k+1} $(k = 1, \ldots, n-1)$.

You can assume that $1 \leq n \leq 1000$.

Output

Print the smallest non-negative integer not appearing in the sequence.

Sample Input 1

```
3
3 0 1
```

Sample Output 1

```
2
```

Sample Input 2

```
11
9 8 7 6 5 4 3 2 1 1 0
```

Sample Output 2

```
12
```

Sample Input 3

```
10
9 0 8 7 6 5 4 3 2 1
```

Sample Output 3

```
10
```

Sample Input 4

```
100
3 6 7 5 3 5 6 2 9 1 2 7 0 9 3 6 0 6 2
6 1 8 7 9 2 0 2 3 7 5 9 2 2 8 9 7 3 6
1 2 9 3 1 9 4 7 8 4 5 0 3 6 1 0 6 3 2
0 6 1 5 5 4 7 6 5 6 9 3 7 4 5 2 5 4 7
4 4 3 0 7 8 6 8 8 4 3 1 4 9 2 0 6 8 9
2 6 6 4 9
```

Sample Output 4

```
11
```

Sample Input 5

```
100
7 2 7 5 4 7 4 4 5 8 1 5 7 7 0 5 6 2 0
4 3 4 1 1 0 6 1 6 6 2 1 7 9 2 4 6 9 3
6 2 8 0 5 9 7 6 3 1 4 9 1 9 1 2 6 4 2
9 7 8 3 9 5 5 2 3 3 8 4 0 6 8 2 5 5 0
6 7 1 8 5 1 4 8 1 3 7 3 3 5 3 0 6 0 6
5 3 2 2 2
```

Sample Output 5

```
86
```

Sample Input 6

```
1
3
```

Sample Output 6

```
0
```

Problem B
Squeeze the Cylinders

Input: Standard Input
Time Limit: 1 second

Laid on the flat ground in the stockyard are a number of heavy metal cylinders with (possibly) different diameters but with the same length. Their ends are aligned and their axes are oriented to exactly the same direction.

We'd like to minimize the area occupied. The cylinders are too heavy to lift up, although rolling them is not too difficult. So, we decided to push the cylinders with two high walls from both sides.

Your task is to compute the minimum possible distance between the two walls when cylinders are squeezed as much as possible. Cylinders and walls may touch one another. They cannot be lifted up from the ground, and thus their order cannot be altered.

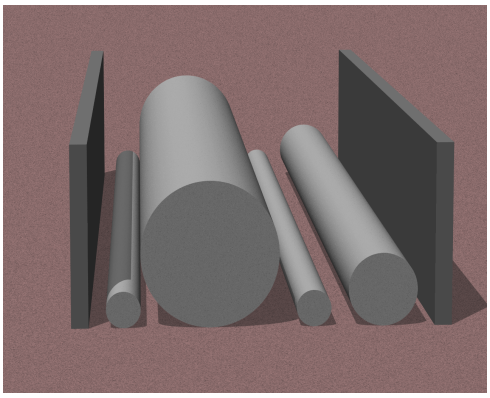


Figure B.1. Cylinders between two walls

Input

The input consists of a single test case. The first line has an integer N ($1 \leq N \leq 500$), which is the number of cylinders. The second line has N positive integers at most 10,000. They are the radii of cylinders from one side to the other.

Output

Print the distance between the two walls when they fully squeeze up the cylinders. The number should not contain an error greater than 0.0001.

Sample Input 1	Sample Output 1
2 10 10	40.00000000

Sample Input 2	Sample Output 2
2 4 12	29.85640646

Sample Input 3	Sample Output 3
5 1 10 1 10 1	40.00000000

Sample Input 4	Sample Output 4
3 1 1 1	6.00000000

Sample Input 5	Sample Output 5
2 5000 10000	29142.13562373

The following figures correspond to the Sample 1, 2, and 3.

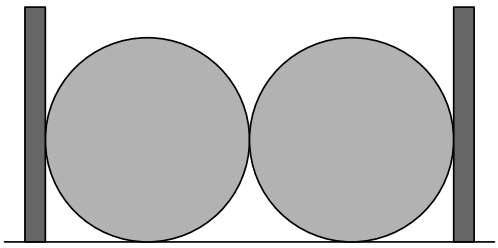


Figure B.2. Sample 1

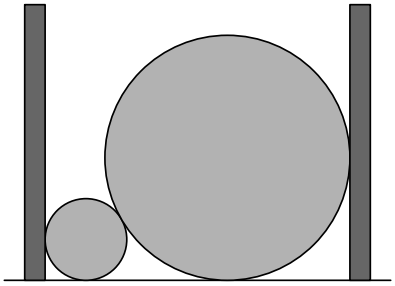


Figure B.3. Sample 2

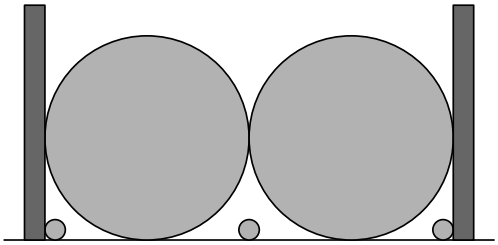


Figure B.4. Sample 3

Problem C
Sibling Rivalry

Input: Standard Input
Time Limit: 2 seconds

You are playing a game with your elder brother.

First, a number of circles and arrows connecting some pairs of the circles are drawn on the ground. Two of the circles are marked as the *start circle* and the *goal circle*.

At the start of the game, you are on the start circle. In each *turn* of the game, your brother tells you a number, and you have to take that number of *steps*. At each step, you choose one of the arrows outgoing from the circle you are on, and move to the circle the arrow is heading to. You can visit the same circle or use the same arrow any number of times.

Your aim is to stop on the goal circle after the fewest possible turns, while your brother's aim is to prevent it as long as possible. Note that, in each single turn, you *must* take the exact number of steps your brother tells you. Even when you visit the goal circle during a turn, you have to leave it if more steps are to be taken.

If you reach a circle with no outgoing arrows before completing all the steps, then you lose the game. You also have to note that, your brother may be able to repeat turns forever, not allowing you to stop after any of them.

Your brother, mean but not too selfish, thought that being allowed to choose arbitrary numbers is not fair. So, he decided to declare three numbers at the start of the game and to use only those numbers.

Your task now is, given the configuration of circles and arrows, and the three numbers declared, to compute the smallest possible number of turns within which you can always finish the game, no matter how your brother chooses the numbers.

Input

The input consists of a single test case, formatted as follows.

$$
\begin{array}{l}
n\ m\ a\ b\ c \\
u_1\ v_1 \\
\vdots \\
u_m\ v_m
\end{array}
$$

All numbers in a test case are integers. n is the number of circles ($2 \leq n \leq 50$). Circles are numbered 1 through n. The start and goal circles are numbered 1 and n, respectively. m is the number of arrows ($0 \leq m \leq n(n-1)$). a, b, and c are the three numbers your brother declared ($1 \leq a, b, c \leq 100$). The pair, u_i and v_i, means that there is an arrow from the circle u_i to the circle v_i. It is ensured that $u_i \neq v_i$ for all i, and $u_i \neq u_j$ or $v_i \neq v_j$ if $i \neq j$.

Output

Print the smallest possible number of turns within which you can always finish the game. Print `IMPOSSIBLE` if your brother can prevent you from reaching the goal, by either making you repeat the turns forever or leading you to a circle without outgoing arrows.

Sample Input 1

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

Sample Output 1

```
IMPOSSIBLE
```

Sample Input 2

```
8 12 1 2 3
1 2
2 3
1 4
2 4
3 4
1 5
5 8
4 6
6 7
4 8
6 8
7 8
```

Sample Output 2

```
2
```

For Sample Input 1, your brother may choose 1 first, then 2, and repeat these forever. Then you can never finish.

For Sample Input 2 (Figure C.1), if your brother chooses 2 or 3, you can finish with a single turn. If he chooses 1, you will have three options.

- Move to the circle 5. This is a bad idea: Your brother may then choose 2 or 3 and make you lose.
- Move to the circle 4. This is the best choice: From the circle 4, no matter any of 1, 2, or 3 your brother chooses in the next turn, you can finish immediately.

- Move to the circle 2. This is not optimal for you. If your brother chooses 1 in the next turn, you cannot finish yet. It will take three or more turns in total.

In summary, no matter how your brother acts, you can finish within two turns. Thus the answer is 2.

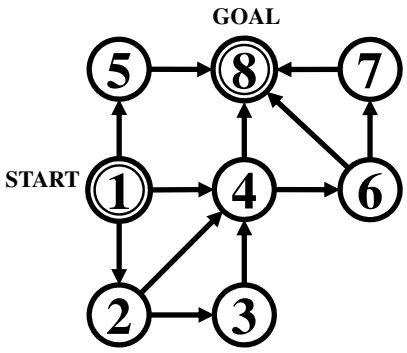


Figure C.1. Sample Input 2

Problem D
Wall Clocks

Input: Standard Input
Time Limit: 1 second

You are the manager of a chocolate sales team. Your team customarily takes tea breaks every two hours, during which varieties of new chocolate products of your company are served. Everyone looks forward to the tea breaks so much that they frequently give a glance at a wall clock.

Recently, your team has moved to a new office. You have just arranged desks in the office. One team member asked you to hang a clock on the wall in front of her desk so that she will not be late for tea breaks. Naturally, everyone seconded her.

You decided to provide an enough number of clocks to be hung in the field of view of everyone. Your team members will be satisfied if they have at least one clock (regardless of the orientation of the clock) in their view, or, more precisely, within 45 degrees left and 45 degrees right (both ends inclusive) from the facing directions of their seats. In order to buy as few clocks as possible, you should write a program that calculates the minimum number of clocks needed to meet everyone's demand.

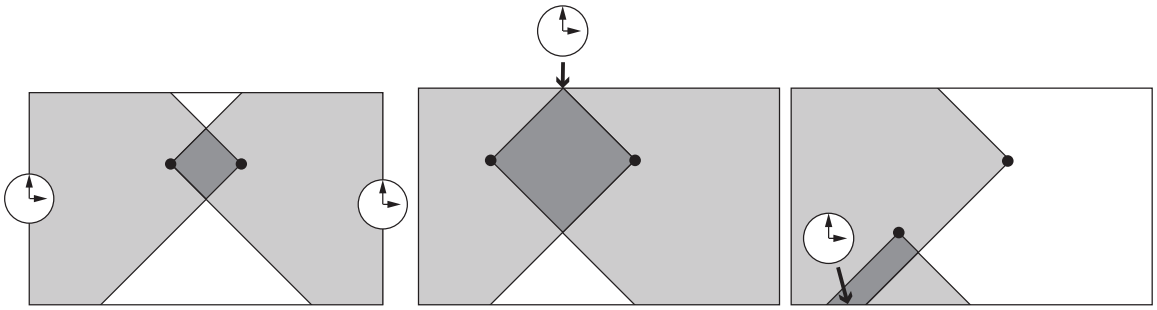
The office room is rectangular aligned to north-south and east-west directions. As the walls are tall enough, you can hang clocks even above the door and can assume one's eyesight is not blocked by other members or furniture. You can also assume that each clock is a point (of size zero), and so you can hang a clock even on a corner of the room.

For example, assume that there are two members. If they are sitting facing each other at positions shown in Figure D.1(A), you need to provide two clocks as they see distinct sections of the wall. If their seats are arranged as shown in Figure D.1(B), their fields of view have a common point on the wall. Thus, you can meet their demands by hanging a single clock at the point. In Figure D.1(C), their fields of view have a common wall section. You can meet their demands with a single clock by hanging it anywhere in the section. Arrangements (A), (B), and (C) in Figure D.1 correspond to Sample Input 1, 2, and 3, respectively.

Input

The input consists of a single test case, formatted as follows.

$$
\begin{array}{l}
n\ w\ d \\
x_1\ y_1\ f_1 \\
\vdots \\
x_n\ y_n\ f_n
\end{array}
$$



(A) (B) (C)

Figure D.1. Arrangements of seats and clocks. Gray area indicates field of view.

All numbers in the test case are integers. The first line contains the number of team members n ($1 \leq n \leq 1,000$) and the size of the office room w and d ($2 \leq w, d \leq 100,000$). The office room has its width w east-west, and depth d north-south. Each of the following n lines indicates the position and the orientation of the seat of a team member. Each member has a seat at a distinct position (x_i, y_i) facing the direction f_i, for $i = 1, \ldots, n$. Here $1 \leq x_i \leq w - 1$, $1 \leq y_i \leq d - 1$, and f_i is one of `N`, `E`, `W`, and `S`, meaning north, east, west, and south, respectively. The position (x, y) means x distant from the west wall and y distant from the south wall.

Output

Print the minimum number of clocks needed.

Sample Input 1

```
2 10 6
4 4 E
6 4 W
```

Sample Output 1

```
2
```

Sample Input 2

```
2 10 6
2 4 E
6 4 W
```

Sample Output 2

```
1
```

Sample Input 3

```
2 10 6
3 2 S
6 4 W
```

Sample Output 3

```
1
```

Sample Input 4

```
6 10 6
1 5 N
7 1 N
8 2 E
9 1 S
4 4 S
3 3 W
```

Sample Output 4

```
3
```

Sample Input 5

```
4 10 6
4 3 W
2 4 N
4 4 W
3 3 S
```

Sample Output 5

```
2
```

Sample Input 6

```
4 100000 40000
25000 25000 S
20000 30000 S
75000 25000 S
80000 30000 S
```

Sample Output 6

```
1
```

Problem E
Bringing Order to Disorder

Input: Standard Input
Time Limit: 1 second

A sequence of digits usually represents a number, but we may define an alternative interpretation. In this problem we define a new interpretation with the order relation $\prec$ among the digit sequences of the same length defined below.

Let s be a sequence of n digits, $d_1 d_2 \cdots d_n$, where each d_i $(1 \leq i \leq n)$ is one of `0`, `1`, ..., and `9`. Let $\mathrm{sum}(s)$, $\mathrm{prod}(s)$, and $\mathrm{int}(s)$ be as follows:

$$\begin{aligned} \mathrm{sum}(s) &= d_1 + d_2 + \cdots + d_n \\ \mathrm{prod}(s) &= (d_1 + 1) \times (d_2 + 1) \times \cdots \times (d_n + 1) \\ \mathrm{int}(s) &= d_1 \times 10^{n-1} + d_2 \times 10^{n-2} + \cdots + d_n \times 10^0 \end{aligned}$$

$\mathrm{int}(s)$ is the integer the digit sequence s represents with normal decimal interpretation.

Let s_1 and s_2 be sequences of the same number of digits. Then $s_1 \prec s_2$ (s_1 is less than s_2) is satisfied if and only if one of the following conditions is satisfied.

1. $\mathrm{sum}(s_1) < \mathrm{sum}(s_2)$
2. $\mathrm{sum}(s_1) = \mathrm{sum}(s_2)$ and $\mathrm{prod}(s_1) < \mathrm{prod}(s_2)$
3. $\mathrm{sum}(s_1) = \mathrm{sum}(s_2)$, $\mathrm{prod}(s_1) = \mathrm{prod}(s_2)$, and $\mathrm{int}(s_1) < \mathrm{int}(s_2)$

For 2-digit sequences, for instance, the following relations are satisfied.

$$\texttt{00} \prec \texttt{01} \prec \texttt{10} \prec \texttt{02} \prec \texttt{20} \prec \texttt{11} \prec \texttt{03} \prec \texttt{30} \prec \texttt{12} \prec \texttt{21} \prec \cdots \prec \texttt{89} \prec \texttt{98} \prec \texttt{99}$$

Your task is, given an n-digit sequence s, to count up the number of n-digit sequences that are less than s in the order $\prec$ defined above.

Input

The input consists of a single test case in a line.

$$d_1 d_2 \cdots d_n$$

n is a positive integer at most 14. Each of $d_1, d_2, \ldots,$ and d_n is a digit.

Output

Print the number of the n-digit sequences less than $d_1 d_2 \cdots d_n$ in the order defined above.

Sample Input 1	Sample Output 1
20	4

Sample Input 2	Sample Output 2
020	5

Sample Input 3	Sample Output 3
118	245

Sample Input 4	Sample Output 4
11111111111111	40073759

Sample Input 5	Sample Output 5
99777222222211	23733362467675

Problem F
Deadlock Detection

Input: Standard Input
Time Limit: 2 seconds

You are working on an analysis of a system with multiple processes and some kinds of resource (such as memory pages, DMA channels, and I/O ports). Each kind of resource has a certain number of instances. A process has to acquire resource instances for its execution. The number of required instances of a resource kind depends on a process. A process finishes its execution and terminates eventually after all the resource in need are acquired. These resource instances are released then so that other processes can use them. No process releases instances before its termination. Processes keep trying to acquire resource instances in need, one by one, without taking account of behavior of other processes. Since processes run independently of others, they may sometimes become unable to finish their execution because of *deadlock.*

A process has to wait when no more resource instances in need are available until some other processes release ones on their termination. Deadlock is a situation in which two or more processes wait for termination of each other, and, regrettably, forever. This happens with the following scenario: One process A acquires the sole instance of resource X, and another process B acquires the sole instance of another resource Y; after that, A tries to acquire an instance of Y, and B tries to acquire an instance of X. As there are no instances of Y other than one acquired by B, A will never acquire Y before B finishes its execution, while B will never acquire X before A finishes. There may be more complicated deadlock situations involving three or more processes.

Your task is, receiving the system's resource allocation time log (from the system's start to a certain time), to determine when the system fell into a *deadlock-unavoidable* state. Deadlock may usually be avoided by an appropriate allocation order, but deadlock-unavoidable states are those in which some resource allocation has already been made and no allocation order from then on can ever avoid deadlock.

Let us consider an example corresponding to Sample Input 1 below. The system has two kinds of resource R_1 and R_2, and two processes P_1 and P_2. The system has three instances of R_1 and four instances of R_2. Process P_1 needs three instances of R_1 and two instances of R_2 to finish its execution, while process P_2 needs a single instance of R_1 and three instances of R_2. The resource allocation time log is given as follows.

time	event	P_1's need R_1	P_1's need R_2	P_2's need R_1	P_2's need R_2	available R_1	available R_2	deadlock
0	start.	3	2	1	3	3	4	
1	P_1 acquired R_1.	2	2	1	3	2	4	
2	P_2 acquired R_2.	2	2	1	2	2	3	
3	P_1 acquired R_2.	2	1	1	2	2	2	
4	P_2 acquired R_1.	2	1	0	2	1	2	avoidable by finishing P_2 first
5	P_1 acquired R_2.	2	0	0	2	1	1	unavoidable
6	P_2 acquired R_2.	2	0	0	1	1	0	
7	P_1 acquired R_1.	1	0	0	1	0	0	arisen

At time 4, P_2 acquired R_1 and the number of available instances of R_1 became less than P_1's need of R_1. Therefore, it became necessary for P_1 to wait P_2 to terminate and release the instance. However, at time 5, P_1 acquired R_2 necessary for P_2 to finish its execution, and thus it became necessary also for P_2 to wait P_1; the deadlock became unavoidable at this time.

Note that the deadlock was still avoidable at time 4 by finishing P_2 first (Sample Input 2).

Input

The input consists of a single test case formatted as follows.

$$
\begin{array}{l}
p\ r\ t \\
l_1 \cdots l_r \\
n_{1,1} \cdots n_{1,r} \\
\vdots \\
n_{p,1} \cdots n_{p,r} \\
P_1\ R_1 \\
\vdots \\
P_t\ R_t
\end{array}
$$

p is the number of processes, and is an integer between 2 and 300, inclusive. The processes are numbered 1 through p. r is the number of resource kinds, and is an integer between 1 and 300, inclusive. The resource kinds are numbered 1 through r. t is the length of the time log, and is an integer between 1 and $200,000$, inclusive. l_j ($1 \leq j \leq r$) is the number of initially available instances of the resource kind j, and is an integer between 1 and 100, inclusive. $n_{i,j}$ ($1 \leq i \leq p$, $1 \leq j \leq r$) is the number of resource instances of the resource kind j that the process i needs, and is an integer between 0 and l_j, inclusive. For every i, at least one of $n_{i,j}$ is non-zero. Each pair of P_k and R_k ($1 \leq k \leq t$) is a resource allocation log at time k meaning that process P_k acquired an instance of resource R_k.

You may assume that the time log is consistent: no process acquires unnecessary resource instances; no process acquires instances after its termination; and a process does not acquire any instance of a resource kind when no instance is available.

Output

Print the time when the system fell into a deadlock-unavoidable state. If the system could still avoid deadlock at time t, print -1.

Sample Input 1

```
2 2 7
3 4
3 2
1 3
1 1
2 2
1 2
2 1
1 2
2 2
1 1
```

Sample Output 1

```
5
```

Sample Input 2

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

Sample Output 2

```
-1
```

Problem G
Do Geese See God?

Input: Standard Input
Time Limit: 3 seconds

A palindrome is a sequence of characters which reads the same backward or forward. Famous ones include "dogeeseseegod" ("Do geese see God?"), "amoreroma" ("Amore, Roma.") and "risetovotesir" ("Rise to vote, sir.").

An ancient sutra has been handed down through generations at a temple on Tsukuba foothills. They say the sutra was a palindrome, but some of the characters have been dropped through transcription.

A famous scholar in the field of religious studies was asked to recover the original. After long repeated deliberations, he concluded that no information to recover the original has been lost, and that the original can be reconstructed as the shortest palindrome including all the characters of the current sutra in the same order. That is to say, the original sutra is obtained by adding some characters before, between, and/or behind the characters of the current.

Given a character sequence, your program should output one of the shortest palindromes containing the characters of the current sutra preserving their order. One of the shortest? Yes, more than one shortest palindromes may exist. Which of them to output is also specified as its rank among the candidates in the lexicographical order.

For example, if the current sutra is `cdba` and 7th is specified, your program should output `cdabadc` among the 8 candidates, `abcdcba`, `abdcdba`, `acbdbca`, `acdbdca`, `cabdbac`, `cadbdac`, `cdabadc` and `cdbabdc`.

Input

The input consists of a single test case. The test case is formatted as follows.

S
k

The first line contains a string S composed of lowercase letters from '`a`' to '`z`'. The length of S is greater than 0, and less than or equal to 2000. The second line contains an integer k which represents the rank in the lexicographical order among the candidates of the original sutra. $1 \leq k \leq 10^{18}$.

Output

Output the k-th string in the lexicographical order among the candidates of the original sutra. If the number of the candidates is less than k, output `NONE`.

Though the first lines of the Sample Input/Output 7 are folded at the right margin, they are actually single lines.

Sample Input 1

```
crc
1
```

Sample Output 1

```
crc
```

Sample Input 2

```
icpc
1
```

Sample Output 2

```
icpci
```

Sample Input 3

```
hello
1
```

Sample Output 3

```
heolloeh
```

Sample Input 4

```
hoge
8
```

Sample Output 4

```
hogegoh
```

Sample Input 5

```
hoge
9
```

Sample Output 5

```
NONE
```

Sample Input 6

```
bbaaab
2
```

Sample Output 6

```
NONE
```

Sample Input 7	Sample Output 7
thdstodxtksrnfacdsohnlfuivqvqsozdstwa szmkboehgcerwxawuojpfuvlxxdfkezprodne ttawsyqazekcftgqbrrtkzngaxzlnphynkmsd sdleqaxnhehwzgzwtldwaacfczqkfpvxnalnn hfzbagzhqhstcymdeijlbkbbubdnptolrmemf xlmmzhfpshykxvzbjmcnsusllpyqghzhdvljd xrrebeef 11469362357953500	feeberrthdstodxtksrnfacdjsohnlfuivdhq vqsozhgdqypllstwausnzcmjkboehgcerzvwx akyhswuojpfhzumvmlxxdfkmezmprlotpndbu bbkblnjiedttmawsyqazekcftgshqbrhrtkzn gaxbzfhnnlanxvphyfnkqmzcsdfscaawdleqa xtnhehwzgzwhehntxaqeldwaacsfdsczmqknf yhpvxnalnnhfzbxagnzktrhrbqhsgtfckezaq yswamttdeijnlbkbbubdnptolrpmzemkfdxxl mvmuzhfpjouwshykaxwvzrecgheobkjmcznsu awtsllpyqdghzosqvqhdviuflnhosjdcafnrs ktxdotsdhtrrebeef

Problem H
Rotating Cutter Bits

Input: Standard Input
Time Limit: 3 seconds

The machine tool technology never stops its development. One of the recent proposals is more flexible lathes in which not only the workpiece but also the cutter bit rotate around parallel axles in synchronization. When the lathe is switched on, the workpiece and the cutter bit start rotating at the same angular velocity, that is, to the same direction and at the same rotational speed. On collision with the cutter bit, parts of the workpiece that intersect with the cutter bit are cut out.

To show the usefulness of the mechanism, you are asked to simulate the cutting process by such a lathe.

Although the workpiece and the cutter bit may have complex shapes, focusing on cross sections of them on a plane perpendicular to the spinning axles would suffice. We introduce an xy-coordinate system on a plane perpendicular to the two axles, in which the center of rotation of the workpiece is at the origin $(0, 0)$, while that of the cutter bit is at $(L, 0)$. You can assume both the workpiece and the cutter bit have polygonal cross sections, not necessarily convex.

Note that, even when this cross section of the workpiece is divided into two or more parts, the workpiece remain undivided on other cross sections.

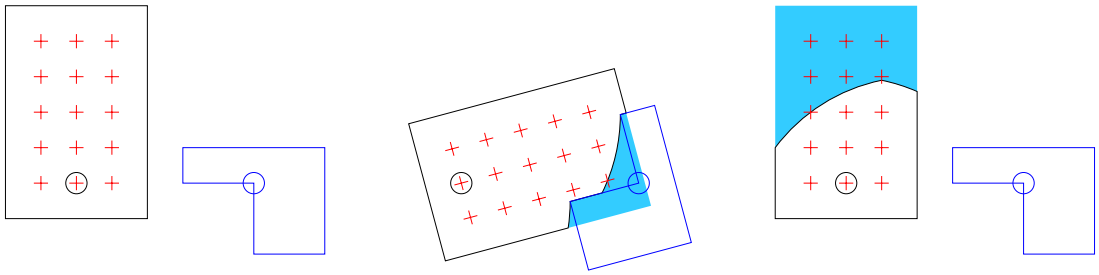
We refer to the lattice points (points with both x and y coordinates being integers) strictly inside, that is, inside and not on an edge, of the workpiece before the rotation as *points of interest*, or POI in short.

Our interest is in how many POI will remain after one full rotation of 360 degrees of both the workpiece and the cutter bit. POI are said to remain if they are strictly inside the resultant workpiece. Write a program that counts them for the given workpiece and cutter bit configuration.

Figure H.1(a) illustrates the workpiece (in black line) and the cutter bit (in blue line) given in Sample Input 1. Two circles indicate positions of the rotation centers of the workpiece and the cutter bit. The red cross-shaped marks indicate the POI.

Figure H.1(b) illustrates the workpiece and the cutter bit in progress in case that the rotation direction is clockwise. The light blue area indicates the area cut-off.

Figure H.1(c) illustrates the result of this sample. Note that one of POI is on the edge of the resulting shape. You should not count this point. There are eight POI remained.



(a) (b) (c)

Figure H.1. The workpiece and the cutter bit in Sample 1

Input

The input consists of a single test case with the following format.

$$
\begin{aligned}
&M\ N\ L\\
&x_{\mathrm{w}1}\ y_{\mathrm{w}1}\\
&\vdots\\
&x_{\mathrm{w}M}\ y_{\mathrm{w}M}\\
&x_{\mathrm{c}1}\ y_{\mathrm{c}1}\\
&\vdots\\
&x_{\mathrm{c}N}\ y_{\mathrm{c}N}
\end{aligned}
$$

The first line contains three integers. M is the number of vertices of the workpiece ($4 \leq M \leq 20$) and N is the number of vertices of the cutter bit ($4 \leq N \leq 20$). L specifies the position of the rotation center of the cutter bit ($1 \leq L \leq 10000$).

Each of the following M lines contains two integers. The i-th line has $x_{\mathrm{w}i}$ and $y_{\mathrm{w}i}$, telling that the position of the i-th vertex of the workpiece has the coordinates $(x_{\mathrm{w}i}, y_{\mathrm{w}i})$. The vertices are given in the counter-clockwise order.

N more following lines are positions of the vertices of the cutter bit, in the same manner, but the coordinates are given as offsets from its center of rotation, $(L, 0)$. That is, the position of the j-th vertex of the cutter bit has the coordinates $(L + x_{\mathrm{c}j}, y_{\mathrm{c}j})$.

You may assume $-10000 \leq x_{\mathrm{w}i}, y_{\mathrm{w}i}, x_{\mathrm{c}j}, y_{\mathrm{c}j} \leq 10000$ for $1 \leq i \leq M$ and $1 \leq j \leq N$.

All the edges of the workpiece and the cutter bit at initial rotation positions are parallel to the x-axis or the y-axis. In other words, for each i ($1 \leq i \leq M$), $x_{\mathrm{w}i} = x_{\mathrm{w}i'}$ or $y_{\mathrm{w}i} = y_{\mathrm{w}i'}$ holds, where $i' = (i \mod M) + 1$. Edges are parallel to the x- and the y-axes alternately. These can also be said about the cutter bit.

You may assume that the cross section of the workpiece forms a simple polygon, that is, no two edges have common points except for adjacent edges. The same can be said about the cutter bit. The workpiece and the cutter bit do not touch or overlap before starting the rotation.

Note that $(0, 0)$ is not always inside the workpiece and $(L, 0)$ is not always inside the cutter bit.

Output

Output the number of POI remaining strictly inside the workpiece.

Sample Input 1	**Sample Output 1**
4 6 5 -2 5 -2 -1 2 -1 2 5 -2 1 -2 0 0 0 0 -2 2 -2 2 1	8

Sample Input 2	**Sample Output 2**
14 14 6000 -3000 3000 -3000 -3000 3000 -3000 3000 -2000 2000 -2000 2000 -1000 1000 -1000 1000 0 0 0 0 1000 -1000 1000 -1000 2000 -2000 2000 -2000 3000 3000 3000 -3000 3000 -3000 2000 -2000 2000 -2000 1000 -1000 1000 -1000 0 0 0 0 -1000 1000 -1000 1000 -2000 2000 -2000 2000 -3000 3000 -3000	6785772

Sample Input 3	Sample Output 3
12 12 11 -50 45 -50 -45 40 -45 40 25 -10 25 -10 -5 0 -5 0 15 30 15 30 -35 -40 -35 -40 45 50 -45 50 45 -40 45 -40 -25 10 -25 10 5 0 5 0 -15 -30 -15 -30 35 40 35 40 -45	966

Sample Input 4	Sample Output 4
20 4 11 -5 5 -5 -10 -4 -10 -4 -1 -3 -1 -3 -10 1 -10 1 -4 0 -4 0 -1 1 -1 1 0 4 0 4 -1 10 -1 10 3 1 3 1 4 10 4 10 5 0 0 3 0 3 3 0 3	64

Problem I
Routing a Marathon Race

Input: Standard Input
Time Limit: 3 seconds

As a member of the ICPC (Ibaraki Committee of Physical Competitions), you are responsible for planning the route of a marathon event held in the City of Tsukuba. A great number of runners, from beginners to experts, are expected to take part.

You have at hand a city map that lists all the street segments suited for the event and all the junctions on them. The race is to start at the junction in front of Tsukuba High, and the goal is at the junction in front of City Hall, both of which are marked on the map.

To avoid congestion and confusion of runners of divergent skills, the route should not visit the same junction twice. Consequently, although the street segments can be used in either direction, they can be included at most once in the route. As the main objective of the event is in recreation and health promotion of citizens, time records are not important and the route distance can be arbitrarily decided.

A number of personnel have to be stationed at every junction *on* the route. Junctions *adjacent to* them, i.e., junctions connected directly by a street segment to the junctions on the route, also need personnel staffing to keep casual traffic from interfering the race. The same number of personnel is required when a junction is on the route and when it is adjacent to one, but different junctions require different numbers of personnel depending on their sizes and shapes, which are also indicated on the map.

The municipal authorities are eager in reducing the costs including the personnel expense for events of this kind. Your task is to write a program that plans a route with the minimum possible number of personnel required and outputs that number.

Input

The input consists of a single test case representing a summary city map, formatted as follows.

$n\ m$
c_1
$\vdots$
c_n
$i_1\ j_1$
$\vdots$
$i_m\ j_m$

The first line of a test case has two positive integers, n and m. Here, n indicates the number of junctions in the map ($2 \leq n \leq 40$), and m is the number of street segments connecting adjacent junctions. Junctions are identified by integers 1 through n.

Then comes n lines indicating numbers of personnel required. The k-th line of which, an integer c_k ($1 \leq c_k \leq 100$), is the number of personnel required for the junction k.

The remaining m lines list street segments between junctions. Each of these lines has two integers i_k and j_k, representing a segment connecting junctions i_k and j_k ($i_k \neq j_k$). There is at most one street segment connecting the same pair of junctions.

The race starts at junction 1 and the goal is at junction n. It is guaranteed that there is at least one route connecting the start and the goal junctions.

Output

Output an integer indicating the minimum possible number of personnel required.

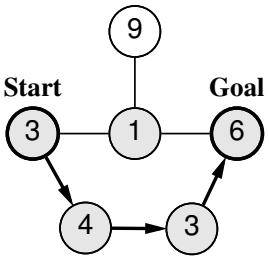


Figure I.1. The Lowest-Cost Route for Sample Input 1

Figure I.1 shows the lowest-cost route for Sample Input 1. The arrows indicate the route and the circles painted gray are junctions requiring personnel assignment. The minimum number of required personnel is 17 in this case.

Sample Input 1	**Sample Output 1**
6 6 3 1 9 4 3 6 1 2 1 4 2 6 5 4 6 5 3 2	17

Problem J
Post Office Investigation

Input: Standard Input
Time Limit: 3 seconds

In this country, all international mails from abroad are first gathered to the central post office, and then delivered to each destination post office relaying some post offices on the way. The delivery routes between post offices are described by a directed graph $G = (V, E)$, where V is the set of post offices and E is the set of possible mail forwarding steps. Due to the inefficient operations, you *cannot* expect that the mails are delivered along the shortest route.

The set of post offices can be divided into a certain number of groups. Here, a group is defined as a set of post offices where mails can be forwarded from any member of the group to any other member, directly or indirectly. The number of post offices in such a group does not exceed 10.

The post offices frequently receive complaints from customers that some mails are not delivered yet. Such a problem is usually due to system trouble in a single post office, but identifying which is not easy. Thus, when such complaints are received, the customer support sends staff to check the system of each candidate post office. Here, the investigation cost to check the system of the post office u is given by c_u, which depends on the scale of the post office.

Since there are many post offices in the country, and such complaints are frequently received, reducing the investigation cost is an important issue. To reduce the cost, the post service administration determined to use the following scheduling rule: When complaints on undelivered mails are received by the post offices $w_1, \ldots, w_k$ one day, staff is sent on the next day to investigate a single post office v with the lowest investigation cost among candidates. Here, the post office v is a candidate if all mails from the central post office to the post offices $w_1, \ldots, w_k$ must go through v. If no problem is found in the post office v, we have to decide the order of investigating other post offices, but the problem is left to some future days.

Your job is to write a program that finds the cost of the lowest-cost candidate when the list of complained post offices in a day, described by $w_1, \ldots, w_k$, is given as a query.

Input

The input consists of a single test case, formatted as follows.

$$
\begin{array}{l}
n\ m \\
u_1\ v_1 \\
\vdots \\
u_m\ v_m \\
c_1 \\
\vdots \\
c_n \\
q \\
k_1\ w_{11}\ \cdots\ w_{1k_1} \\
\vdots \\
k_q\ w_{q1}\ \cdots\ w_{qk_q}
\end{array}
$$

n is the number of post offices ($2 \leq n \leq 50,000$), which are numbered from 1 to n. Here, post office 1 corresponds to the central post office. m is the number of forwarding pairs of post offices ($1 \leq m \leq 100,000$). The pair, u_i and v_i, means that some of the mails received at post office u_i are forwarded to post office v_i ($i = 1, \ldots, m$). c_j is the investigation cost for the post office j ($j = 1, \ldots, n$, $1 \leq c_j \leq 10^9$). q ($q \geq 1$) is the number of queries, and each query is specified by a list of post offices which received undelivered mail complaints. k_i ($k_i \geq 1$) is the length of the list and $w_{i1}, \ldots, w_{ik_i}$ are the distinct post offices in the list. $\sum_{i=1}^{q} k_i \leq 50,000$.

You can assume that there is at least one delivery route from the central post office to all the post offices.

Output

For each query, you should output a single integer that is the lowest cost of the candidate of troubled post office.

Sample Input 1	Sample Output 1
8 8 1 2 1 3 2 4 2 5 2 8 3 5 3 6 4 7 1000 100 100 10 10 10 1 1 3 2 8 6 2 4 7 2 7 8	1000 10 100

Sample Input 2	Sample Output 2
10 12 1 2 2 3 3 4 4 2 4 5 5 6 6 7 7 5 7 8 8 9 9 10 10 8 10 9 8 7 6 5 4 3 2 1 3 2 3 4 3 6 7 8 3 9 6 3	8 5 8

Problem K
Min-Max Distance Game

Input: Standard Input
Time Limit: 1 second

Alice and Bob are playing the following game. Initially, n stones are placed in a straight line on a table. Alice and Bob take turns alternately. In each turn, the player picks one of the stones and removes it. The game continues until the number of stones on the straight line becomes two. The two stones are called *result stones*. Alice's objective is to make the result stones as distant as possible, while Bob's is to make them closer.

You are given the coordinates of the stones and the player's name who takes the first turn. Assuming that both Alice and Bob do their best, compute and output the distance between the result stones at the end of the game.

Input

The input consists of a single test case with the following format.

$$n\ f$$
$$x_1\ x_2\ \cdots\ x_n$$

n is the number of stones ($3 \leq n \leq 10^5$). f is the name of the first player, either `Alice` or `Bob`. For each i, x_i is an integer that represents the distance of the i-th stone from the edge of the table. It is guaranteed that $0 \leq x_1 < x_2 < \cdots < x_n \leq 10^9$ holds.

Output

Output the distance between the result stones in one line.

Sample Input 1

```
5 Alice
10 20 30 40 50
```

Sample Output 1

```
30
```

Sample Input 2

```
5 Bob
2 3 5 7 11
```

Sample Output 2

```
2
```