2018 LGE Code Jam: Online Round 1

Problem A. Crossword

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 초
Memory limit:	512 MB

$N \times M$ sized table contains capital letters in alphabet, in each space. Write a program that can decide whether there is a word S when given.

If S exists in the table, starting from one space every letter should come along in order. Note that up, down, left, right and diagonal direction are all possible. Diagonal direction means upper left, upper right, lower left and lower right. The direction cannot be shifted on its way.

Input

In the first line of input, a word S which has less than or equal to 100 letters is written. S only contains capital letters.

In the second line of input, N and M which are the numbers of rows and columns. N and M are integers which are less than or equal to 100.

Starting from the third line to $N+2$-th line of input, the alphabet in each row are written.

Output

Print 1 if there is a word S in the given table. Otherwise, print 0.

Examples

standard input	standard output
ABCD 5 5 ACDBE ABCED ACCEE ACHDF ACBCE	1

Notes

In Input Example #1, word ABCD comes like below.

```
ACDBE
ABCED
ACCEE
ACHDF
ACBCE
```

In Input Example #2, word STR can be seen very often. Only some of them are bold faced below.

```
STARTS
STRSTR
RRTSRE
SRSTRR
STRTSR
STSTSS
```

Problem B. Reverse

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

There is a sequence made out of N numbers. The index of sequence starts from 1. We want to apply an operation M times. Write a program which can determine where the K-th number goes to, after all the operations ends.

The operation consists of one integer i.

If i is a positive number, reverse the order of i number of the sequence from the beginning. If i is a negative number, reverse the order of $-i$ number of the sequence from the end.

For example, when $N = 5$, $A = 1, 3, 2, 4, 5$ and operation 3 is applied, sequence A becomes 2, 3, 1, 4, 5. Then if operation -4 is applied, it becomes 2, 5, 4, 1, 3. The 1st number at the beginning goes to 4th, and 3rd number goes to 1st.

Input

In the first line of input, size $N(1 \leq N \leq 100,000)$ of sequence A, location $K(1 \leq K \leq N)$, and the number of operation $M(1 \leq M \leq 100,000)$ is written. In the second line of input, numbers in sequence A are written starting from the 1st number in order. Numbers in the sequence are natural numbers which are less than or equal to 100,000. Same numbers can be contained several times in the sequence.

Starting from the third line of input, integer i is written one at a time. The absolute value of i is less than or equal to N and is not 0.

Output

Print in the first row where the K-th number goes to after M times of operations are complete.

Examples

standard input	standard output
5 1 2 1 3 2 4 5 3 -4	4
5 3 2 5 5 5 5 5 3 -4	1
4 4 8 10 20 15 25 1 -2 3 -4 -1 2 -3 4	3
10 1 6 9 8 1 2 8 7 3 4 6 6 9 -9 8 -8 7 -7	1
5 2 5 7 6 5 4 3 3 -3 3 -3 3	2
5 3 2 5 5 5 5 5 4 -3	2

Problem C. Dice Tower

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

Sally has N number of dices. All the dices are a cube figure with same size. Unlike normal dice, one number is written on each face of a cube. Each cube can contain only one number.

A dice tower means a figure of stacked dices. Dices must be stacked in one row; one dice on top of another. She wants to minimize the number of dice towers made out of N number of dices.

If the number written on the dice is called s, maximum number of dices which can be stacked on the dice is s.

For instant, let's think of a case where there are 4 dices and each number written on them are 1, 2, 4, 5. The result 2, 1, 4, 5 from the top is possible. Because there are 0 dices on a dice which is written 2, 1 dice on a dice written 1, 2 dices on a dice written 4, and 3 dices on a dice written 5. But order 4, 1, 5, 2 is impossible. Because there are total 3 dices on a dice written 2.

Write a program that can get the minimum number of dice towers, when N and number written on each dices are given.

Input

In the first line of input, the number of dices $N (1 \leq N \leq 1,000)$ is written. In the second line, numbers written on each dices are written. Number written on a dice is 0 or a positive number less than or equal to 1,000, and sorted by blank space.

Output

Print the mininum number of dice towers in the first line.

Examples

standard input	standard output
4 1 2 4 5	1
4 1 2 1 2	2
5 0 0 0 0 0	5

Notes

A dice tower is described as $(s_1, s_2, \ldots, s_k)$, when k number of dices are stacked and number written on each dice are $s_1, s_2, \ldots, s_k$ from the top.

In Example #1 we can make one dice tower as (1, 2, 4, 5) or (2, 1, 4, 5), and there are other ways too.

In Example #2 we cannot make less than two dice towers. (1, 2), (1, 2) makes two. (1, 1, 2), (2) is also possible.

In Example #3 every number written on the dice are 0. Therefore no dices can be stacked on a dice. Hence every dice tower is made out of one dice.

Problem D. LG Marathon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

LG marathon is coming. People must run through a predeterminated course, which is not determinated yet. There are N number of cross streets and M number of two-way streets where the marathon takes place. Two-way street connects two cross streets. Cross streets are numbered from 1 to N, and two-way streets are numbered from 1 to M.

Marathon course can be represented as $(V_1, V_2, \ldots, V_k)$, where k is the number of cross streets included in the course. The first cross street V_1 is always cross street #1, and the last one V_k is always cross street #N. Same cross street cannot come more than one time, and each cross street must be connected by two-way street.

To hold a marathon we must control the street. We may or may not pay for controlling the street. Street control cost C and upper limit of free pass T exist on each street. If P is the number of participants, street can be controlled without any payment when $P \leq T$. If $P > T$, the cost is $C \times (P - T)^2$.

Maximum cost in budget that can be paid during the marathon is K. We want to maximize the participants. Print the number of maximum participants when the course is determined appropriately within budget.

Input

In the first line of input, the number of cross street N, the number of two-way street M, and budget K is written. ($2 \leq N \leq 100,000$, $N - 1 \leq M \leq 100,000$, $1 \leq K \leq 10^9$)

From the second line to M-th line of input, information of streets are written. The information consists of four numbers A, B, C, T ($1 \leq A < B \leq N$, $1 \leq C, T \leq 1,000$). A and B are the number of two cross streets connected by a two-way street, C and T are used in calculating the street control cost.

Maximun number of two-way street connecting two cross streets is 1, and a case that has an answer must be written in input always.

Output

Print the maximum number of participants, when the course is determined within budget, in the first line .

Subtask 1 (8 points)

- $2 \leq N \leq 1,000$
- $N - 1 \leq M \leq 100,000$
- $1 \leq K \leq 1,000$

Subtask 2 (14 points)

- $2 \leq N \leq 1,000$
- $N - 1 \leq M \leq 100,000$
- $1 \leq K \leq 10^9$

Subtask 3 (12 points)

- $2 \leq N \leq 100,000$
- $N - 1 \leq M \leq 100,000$
- $1 \leq K \leq 10^9$

Examples

standard input	standard output
3 3 5 1 2 1 1 1 3 1 1 2 3 1 1	3
3 3 3 1 2 1 1 1 3 1 1 2 3 1 1	2
3 2 25 1 2 5 1 2 3 1 5	3
4 5 100 1 2 3 4 1 3 1 2 2 3 2 1 3 4 1 1 2 4 1 5	9

Notes

In Example #1, when the course is (1, 3), 3 people can participate. On this occasion street control cost connecting 1 and 3 is $1 \times (3 - 1)^2 = 4$. If 4 people participates, the cost becomes $1 \times (4 - 1)^2 = 9$ which is off the budget.

Example #2 is the same to #1 but the budget is now 3. Therefore 3 people cannot participate. On this occasion, if 2 people participates the cost becomes $1 \times (2 - 1)^2 = 1$.

In Example #3, the course which starts in cross street #1 and ends in cross street #3 is only (1, 2, 3). When we calculate the number of participants and control cost, results are below.

- 1 person participates: $5 \times (1 - 1)^2 + 0 = 0$
- 2 people participates: $5 \times (2 - 1)^2 + 0 = 5 + 0 = 5$
- 3 people participates: $5 \times (3 - 1)^2 + 0 = 20 + 0 = 20$
- 4 people participates: $5 \times (4 - 1)^2 + 0 = 45 + 0 = 45$
- 5 people participates: $5 \times (5 - 1)^2 + 0 = 80 + 0 = 80$
- 6 people participates: $5 \times (6 - 1)^2 + 1 \times (6 - 5)^2 = 125 + 1 = 126$
- 7 people participates: $5 \times (7 - 1)^2 + 1 \times (7 - 5)^2 = 180 + 4 = 184$

With budget 25, maximum 3 people can participate.

Problem E. Block Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

There are N numbers of blocks in a row. The one the leftend is numbered 1 and the one next to it is numbered 2. So the one on the rightend is numbered N. The height of the i-th block is H_i.

To play the block game, we must put a little machine in front of a block. At first this machine is in front of block #1. The object of this game is to eliminate every blocks using the machine. In this case, the sequence of the height in eliminated order must be in non-decreasing order.

The machine can perform three commands below.

- Move to the right block. If the machine was in front of i-th block, move to in front of $i+1$-th block. If there is no block on the right, the command cannot be performed.
- Move to the left block. If the machine was in front of i-th block, move to in front of $i-1$-th block. If there is no block on the left, this command cannot be performed.
- the block behind the machine and move to the left or right. The order must include the direction after the elimination. If every block are gone it doesn't have to move.

The digit of blocks change after the elimination command. For instance, the height of the blocks are (2, 3, 4, 5, 6) and the machine is in front of the block with height 4. The block with height 4 is the third one from the left so it is #3. If the machine eliminates this block and move to the left, the height becomes (2, 3, 5, 6) and the machine is in front of the block with height 3(block #2). If it moves to the right, the height becomes (2, 3, 5, 6) and it is in front of the block with height 5(block #3).

Write a program which calculates the minimum numbers of commands to achive the object of this game.

If a sequence $A_1, A_2, \ldots, A_k$ satisfies $A_1 \leq A_2 \leq ... \leq A_k$, we call it non-decreasing order.

Input

In the first line of input, the number of blocks $N (1 \leq N \leq 100,000)$ is written. In the second line, the height of each blocks H_i are written in order. $(1 \leq H_i \leq 100,000)$

Output

Print the minimum numbers of commands to achive the object of this game in the first line.

Subtask 1 (7 points)

- $1 \leq N \leq 1,000$
- $1 \leq H_i \leq 1,000$
- H_i는 중복되지 않음

Subtask 2 (14 points)

- $1 \leq N \leq 1,000$
- $1 \leq H_i \leq 100,000$
- H_i는 중복되지 않음

Subtask 3 (20 points)

- $1 \leq N \leq 100,000$
- $1 \leq H_i \leq 100,000$

Examples

standard input	standard output
3 1 2 3	3
4 4 2 1 3	6
5 5 5 1 5 5	7
10 10 8 6 4 2 1 3 5 7 9	15
10 1 3 5 7 9 10 8 6 4 2	46

Notes

In Example #1, the command should be like below. The machine is in front of the underlined block.

- Initial condition: 1 2 3
- Eliminate the block and move to the right: 2 3
- Eliminate the block and move to the right: 3
- Eliminate the block

In Example #2, the command should be like below.

- Initial condition: 4 2 1 3
- Move to the right: 4 2 1 3
- Move to the right: 4 2 1 3
- Eliminate the block and move to the left: 4 2 3
- Eliminate the block and move to the right: 4 3
- Eliminate the block and move to the left: 4
- Eliminate the block