2017-2018 ACM-ICPC, NEERC, Southern Subregional Contest (Online Mirror, ACM-ICPC Rules, Teams Preferred)

A. Automatic Door

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is an automatic door at the entrance of a factory. The door works in the following way:

- when one or several people come to the door and it is closed, the door immediately opens automatically and all people immediately come inside,
- when one or several people come to the door and it is open, all people immediately come inside,
- opened door immediately closes in d seconds after its opening,
- if the door is closing and one or several people are coming to the door at the same moment, then all of them will have enough time to enter and only after that the door will close.

For example, if $d = 3$ and four people are coming at four different moments of time $t_1 = 4$, $t_2 = 7$, $t_3 = 9$ and $t_4 = 13$ then the door will open three times: at moments 4, 9 and 13. It will close at moments 7 and 12.

It is known that n employees will enter at moments a, $2 \cdot a$, $3 \cdot a$, ..., $n \cdot a$ (the value a is positive integer). Also m clients will enter at moments t_1, t_2, ..., t_m.

Write program to find the number of times the automatic door will open. Assume that the door is initially closed.

Input

The first line contains four integers n, m, a and d ($1 \le n, a \le 10^9$, $1 \le m \le 10^5$, $1 \le d \le 10^{18}$) — the number of the employees, the number of the clients, the moment of time when the first employee will come and the period of time in which the door closes.

The second line contains integer sequence t_1, t_2, ..., t_m ($1 \le t_i \le 10^{18}$) — moments of time when clients will come. The values t_i are given in non-decreasing order.

Output

Print the number of times the door will open.

Examples

input
1 1 3 4 7

output
1

input
4 3 4 2 7 9 11

output
4

Note

In the first example the only employee will come at moment 3. At this moment the door will open and will stay open until the moment 7. At the same moment of time the client will come, so at first he will enter and only after it the door will close. Thus the door will open one time.

B. Berland Army

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n military men in the Berland army. Some of them have given orders to other military men by now. Given m pairs (x_i, y_i), meaning that the military man x_i gave the i-th order to another military man y_i.

It is time for reform! The Berland Ministry of Defence plans to introduce ranks in the Berland army. Each military man should be assigned a rank — integer number between 1 and k, inclusive. Some of them have been already assigned a rank, but the rest of them should get a rank soon.

Help the ministry to assign ranks to the rest of the army so that:

- for each of m orders it is true that the rank of a person giving the order (military man x_i) is strictly greater than the rank of a person receiving the order (military man y_i);
- for each rank from 1 to k there is at least one military man with this rank.

Input

The first line contains three integers n, m and k ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$, $1 \le k \le 2 \cdot 10^5$) — number of military men in the Berland army, number of orders and number of ranks.

The second line contains n integers $r_1, r_2, ..., r_n$, where $r_i > 0$ (in this case $1 \le r_i \le k$) means that the i-th military man has been already assigned the rank r_i; $r_i = 0$ means the i-th military man doesn't have a rank yet.

The following m lines contain orders one per line. Each order is described with a line containing two integers x_i, y_i ($1 \le x_i, y_i \le n$, $x_i \ne y_i$). This line means that the i-th order was given by the military man x_i to the military man y_i. For each pair (x, y) of military men there could be several orders from x to y.

Output

Print n integers, where the i-th number is the rank of the i-th military man. If there are many solutions, print any of them.

If there is no solution, print the only number `-1`.

Examples

input

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

output

```
1 3 3 2 2
```

input

```
7 6 5
0 4 5 4 1 0 0
6 1
3 6
3 1
7 5
7 1
7 4
```

output

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

input

```
2 2 2
2 1
1 2
2 1
```

output

```
-1
```

C. Downloading B++

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Only T milliseconds left before the start of well-known online programming contest Codehorses Round 2017.

Polycarp needs to download B++ compiler to take part in the contest. The size of the file is f bytes.

Polycarp's internet tariff allows to download data at the rate of one byte per t_0 milliseconds. This tariff is already prepaid, and its use does not incur any expense for Polycarp. In addition, the Internet service provider offers two additional packages:

- download a_1 bytes at the rate of one byte per t_1 milliseconds, paying p_1 burles for the package;
- download a_2 bytes at the rate of one byte per t_2 milliseconds, paying p_2 burles for the package.

Polycarp can buy any package many times. When buying a package, its price (p_1 or p_2) is prepaid before usage. Once a package is bought it replaces the regular tariff until package data limit is completely used. After a package is consumed Polycarp can immediately buy a new package or switch to the regular tariff without loosing any time. While a package is in use Polycarp can't buy another package or switch back to the regular internet tariff.

Find the minimum amount of money Polycarp has to spend to download an f bytes file no more than in T milliseconds.

Note that because of technical reasons Polycarp can download only integer number of bytes using regular tariff and both packages. I.e. in each of three downloading modes the number of downloaded bytes will be integer. It means that Polycarp can't download a byte partially using the regular tariff or/and both packages.

Input

The first line contains three integer numbers f, T and t_0 ($1 \le f, T, t_0 \le 10^7$) — size of the file to download (in bytes), maximal time to download the file (in milliseconds) and number of milliseconds to download one byte using the regular internet tariff.

The second line contains a description of the first additional package. The line contains three integer numbers a_1, t_1 and p_1 ($1 \le a_1, t_1, p_1 \le 10^7$), where a_1 is maximal sizes of downloaded data (in bytes), t_1 is time to download one byte (in milliseconds), p_1 is price of the package (in burles).

The third line contains a description of the second additional package. The line contains three integer numbers a_2, t_2 and p_2 ($1 \le a_2, t_2, p_2 \le 10^7$), where a_2 is maximal sizes of downloaded data (in bytes), t_2 is time to download one byte (in milliseconds), p_2 is price of the package (in burles).

Polycarp can buy any package many times. Once package is bought it replaces the regular tariff until package data limit is completely used. While a package is in use Polycarp can't buy another package or switch back to the regular internet tariff.

Output

Print the minimum amount of money that Polycarp needs to pay to download B++ compiler no more than in T milliseconds. If there is no solution, print the only integer `-1`.

Examples

input

```
120 964 20
26 8 8
13 10 4
```

output

```
40
```

input

```
10 200 20
1 1 1
2 2 3
```

output

```
0
```

input

```
8 81 11
4 10 16
3 10 12
```

output

```
28
```

input

```
8 79 11
4 10 16
```

```
3 10 12
```

output

```
-1
```

Note

In the first example Polycarp has to buy the first additional package 5 times and do not buy the second additional package. He downloads 120 bytes (of total $26 \cdot 5 = 130$ bytes) in $120 \cdot 8 = 960$ milliseconds ($960 \le 964$). He spends $8 \cdot 5 = 40$ burles on it.

In the second example Polycarp has enough time to download 10 bytes. It takes $10 \cdot 20 = 200$ milliseconds which equals to upper constraint on download time.

In the third example Polycarp has to buy one first additional package and one second additional package.

In the fourth example Polycarp has no way to download the file on time.

D. Packmen Strike Back

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Game field is represented by a line of n square cells. In some cells there are packmen, in some cells there are asterisks and the rest of the cells are empty. Packmen eat asterisks.

Before the game starts you can choose a movement direction, left or right, for each packman. Once the game begins all the packmen simultaneously start moving according their directions. A packman can't change the given direction.

Once a packman enters a cell containing an asterisk, packman immediately eats the asterisk. Once the packman leaves the cell it becomes empty. Each packman moves at speed 1 cell per second. If a packman enters a border cell, the packman stops. Packmen do not interfere with the movement of other packmen; in one cell there can be any number of packmen moving in any directions.

Your task is to assign a direction to each packman so that they eat the maximal number of asterisks. If there are multiple ways to assign directions to eat the maximal number of asterisks, you should choose the way which minimizes the time to do that.

Input

The first line contains integer number n ($2 \le n \le 1\,000\,000$) — the number of cells in the game field.

The second line contains n characters. If the i-th character is '.', the i-th cell is empty. If the i-th character is '*', the i-th cell contains an asterisk. If the i-th character is 'P', the i-th cell contains a packman.

The field contains at least one asterisk and at least one packman.

Output

Print two integer numbers — the maximal number of asterisks packmen can eat and the minimal time to do it.

Examples

input
6 *.P*P*

output
3 4

input
8 *...P..*

output
1 3

Note

In the first example the leftmost packman should move to the right, the rightmost packman should move to the left. All the asterisks will be eaten, the last asterisk will be eaten after 4 seconds.

E. Field of Wonders

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarpus takes part in the "Field of Wonders" TV show. The participants of the show have to guess a hidden word as fast as possible. Initially all the letters of the word are hidden.

The game consists of several turns. At each turn the participant tells a letter and the TV show host responds if there is such letter in the word or not. If there is such letter then the host reveals **all** such letters. For example, if the hidden word is "abacaba" and the player tells the letter "a", the host will reveal letters at all positions, occupied by "a": 1, 3, 5 and 7 (positions are numbered from left to right starting from 1).

Polycarpus knows m words of exactly the same length as the hidden word. The hidden word is also known to him and appears as one of these m words.

At current moment a number of turns have already been made and some letters (possibly zero) of the hidden word are already revealed. Previously Polycarp has told exactly the letters which are currently revealed.

It is Polycarpus' turn. He wants to tell a letter in such a way, that the TV show host will assuredly reveal at least one more letter. Polycarpus cannot tell the letters, which are already revealed.

Your task is to help Polycarpus and find out the number of letters he can tell so that the show host will assuredly reveal at least one of the remaining letters.

Input

The first line contains one integer n ($1 \le n \le 50$) — the length of the hidden word.

The following line describes already revealed letters. It contains the string of length n, which consists of lowercase Latin letters and symbols "*". If there is a letter at some position, then this letter was already revealed. If the position contains symbol "*", then the letter at this position has not been revealed yet. It is guaranteed, that at least one letter is still closed.

The third line contains an integer m ($1 \le m \le 1000$) — the number of words of length n, which Polycarpus knows. The following m lines contain the words themselves — n-letter strings of lowercase Latin letters. All words are distinct.

It is guaranteed that the hidden word appears as one of the given m words. Before the current move Polycarp has told exactly the letters which are currently revealed.

Output

Output the single integer — the number of letters Polycarpus can tell so that the TV show host definitely reveals at least one more letter. It is possible that this number is zero.

Examples

input
4 a**d 2 abcd acbd

output
2

input
5 lo*er 2 lover loser

output
0

input
3 a*a 2 aaa aba

output
1

Note

In the first example Polycarpus can tell letters "b" and "c", which assuredly will be revealed.

The second example contains no letters which can be told as it is not clear, which of the letters "v" or "s" is located at the third position of the hidden word.

In the third example Polycarpus exactly knows that the hidden word is "aba", because in case it was "aaa", then the second letter "a" would have already been revealed in one of previous turns.

F. Lost in Transliteration

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are some ambiguities when one writes Berland names with the letters of the Latin alphabet.

For example, the Berland sound *u* can be written in the Latin alphabet as "u", and can be written as "oo". For this reason, two words "ulyana" and "oolyana" denote the same name.

The second ambiguity is about the Berland sound *h*: one can use both "h" and "kh" to write it. For example, the words "mihail" and "mikhail" denote the same name.

There are *n* users registered on the Polycarp's website. Each of them indicated a name represented by the Latin letters. How many distinct names are there among them, if two ambiguities described above are taken into account?

Formally, we assume that two words denote the same name, if using the replacements "u" $\Leftrightarrow$ "oo" and "h" $\Leftrightarrow$ "kh", you can make the words equal. One can make replacements in both directions, in any of the two words an arbitrary number of times. A letter that resulted from the previous replacement can participate in the next replacements.

For example, the following pairs of words denote the same name:

- "koouper" and "kuooper". Making the replacements described above, you can make both words to be equal: "koouper" $\rightarrow$ "kuuper" and "kuooper" $\rightarrow$ "kuuper".
- "khun" and "kkkhoon". With the replacements described above you can make both words to be equal: "khun" $\rightarrow$ "khoon" and "kkkhoon" $\rightarrow$ "kkhoon" $\rightarrow$ "khoon".

For a given list of words, find the minimal number of groups where the words in each group denote the same name.

Input

The first line contains integer number n ($2 \le n \le 400$) — number of the words in the list.

The following n lines contain words, one word per line. Each word consists of only lowercase Latin letters. The length of each word is between 1 and 20 letters inclusive.

Output

Print the minimal number of groups where the words in each group denote the same name.

Examples

input

```
10
mihail
oolyana
kooooper
hoon
ulyana
koouper
mikhail
khun
kuooper
kkkhoon
```

output

```
4
```

input

```
9
hariton
hkariton
buoi
kkkhariton
boooi
bui
khariton
boui
boi
```

output

```
5
```

input

```
2
alex
alex
```

output

```
1
```

Note

There are four groups of words in the first example. Words in each group denote same name:

1. "mihail", "mikhail"
2. "oolyana", "ulyana"
3. "kooooper", "koouper"
4. "hoon", "khun", "kkkhoon"

There are five groups of words in the second example. Words in each group denote same name:

1. "hariton", "kkkhariton", "khariton"
2. "hkariton"
3. "buoi", "boooi", "boui"
4. "bui"
5. "boi"

In the third example the words are equal, so they denote the same name.

G. Orientation of Edges

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya has a graph containing both directed (oriented) and undirected (non-oriented) edges. There can be multiple edges between a pair of vertices.

Vasya has picked a vertex s from the graph. Now Vasya wants to create two separate plans:

1. to orient each undirected edge in one of two possible directions to *maximize* number of vertices reachable from vertex s;
2. to orient each undirected edge in one of two possible directions to *minimize* number of vertices reachable from vertex s.

In each of two plans each undirected edge must become directed. For an edge chosen directions can differ in two plans.

Help Vasya find the plans.

Input

The first line contains three integers n, m and s ($2 \le n \le 3 \cdot 10^5$, $1 \le m \le 3 \cdot 10^5$, $1 \le s \le n$) — number of vertices and edges in the graph, and the vertex Vasya has picked.

The following m lines contain information about the graph edges. Each line contains three integers t_i, u_i and v_i ($1 \le t_i \le 2$, $1 \le u_i, v_i \le n$, $u_i \ne v_i$) — edge type and vertices connected by the edge. If $t_i = 1$ then the edge is directed and goes from the vertex u_i to the vertex v_i. If $t_i = 2$ then the edge is undirected and it connects the vertices u_i and v_i.

It is guaranteed that there is at least one undirected edge in the graph.

Output

The first two lines should describe the plan which maximizes the number of reachable vertices. The lines three and four should describe the plan which minimizes the number of reachable vertices.

A description of each plan should start with a line containing the number of reachable vertices. The second line of a plan should consist of f symbols '+' and '-', where f is the number of undirected edges in the initial graph. Print '+' as the j-th symbol of the string if the j-th undirected edge (u, v) from the input should be oriented from u to v. Print '-' to signify the opposite direction (from v to u). Consider undirected edges to be numbered in the same order they are given in the input.

If there are multiple solutions, print any of them.

Examples

input

```
2 2 1
1 1 2
2 2 1
```

output

```
2
-
2
+
```

input

```
6 6 3
2 2 6
1 4 5
2 3 4
1 4 1
1 3 1
2 2 3
```

output

```
6
++-
2
+-+
```

H. Palindromic Cut

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Kolya has a string s of length n consisting of lowercase and uppercase Latin letters and digits.

He wants to rearrange the symbols in s and cut it into the minimum number of parts so that each part is a palindrome and all parts have *the same lengths*. A palindrome is a string which reads the same backward as forward, such as `madam` or `racecar`.

Your task is to help Kolya and determine the minimum number of palindromes of equal lengths to cut s into, if it is allowed to rearrange letters in s before cuttings.

Input

The first line contains an integer n ($1 \le n \le 4 \cdot 10^5$) — the length of string s.

The second line contains a string s of length n consisting of lowercase and uppercase Latin letters and digits.

Output

Print to the first line an integer k — minimum number of palindromes into which you can cut a given string.

Print to the second line k strings — the palindromes themselves. Separate them by a space. You are allowed to print palindromes in arbitrary order. All of them should have the same length.

Examples

input
6 aabaac

output
2 aba aca

input
8 0rTrT022

output
1 02TrrT20

input
2 aA

output
2 a A

I. Photo Processing

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Evlampiy has found one more cool application to process photos. However the application has certain limitations.

Each photo i has a contrast v_i. In order for the processing to be truly of high quality, the application must receive at least k photos with contrasts which differ as little as possible.

Evlampiy already knows the contrast v_i for each of his n photos. Now he wants to split the photos into groups, so that each group contains at least k photos. As a result, each photo must belong to exactly one group.

He considers a processing time of the j-th group to be the difference between the maximum and minimum values of v_i in the group. Because of multithreading the processing time of a division into groups is the maximum processing time among all groups.

Split n photos into groups in a such way that the processing time of the division is the minimum possible, i.e. that the the maximum processing time over all groups as least as possible.

Input

The first line contains two integers n and k ($1 \le k \le n \le 3 \cdot 10^5$) — number of photos and minimum size of a group.

The second line contains n integers $v_1, v_2, \ldots, v_n$ ($1 \le v_i \le 10^9$), where v_i is the contrast of the i-th photo.

Output

Print the minimal processing time of the division into groups.

Examples

input
5 2 50 110 130 40 120

output
20

input
4 1 2 3 4 1

output
0

Note

In the first example the photos should be split into 2 groups: $[40, 50]$ and $[110, 120, 130]$. The processing time of the first group is 10, and the processing time of the second group is 20. Maximum among 10 and 20 is 20. It is impossible to split the photos into groups in a such way that the processing time of division is less than 20.

In the second example the photos should be split into four groups, each containing one photo. So the minimal possible processing time of a division is 0.

J. Renovation

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The mayor of the Berland city S sees the beauty differently than other city-dwellers. In particular, he does not understand at all, how antique houses can be nice-looking. So the mayor wants to demolish all ancient buildings in the city.

The city S is going to host the football championship very soon. In order to make the city beautiful, every month the Berland government provides mayor a money tranche. The money has to be spent on ancient buildings renovation.

There are n months before the championship and the i-th month tranche equals to a_i burles. The city S has m antique buildings and the renovation cost of the j-th building is b_j burles.

The mayor has his own plans for spending the money. As he doesn't like antique buildings he wants to demolish as much of them as possible. For the j-th building he calculated its demolishing cost p_j.

The mayor decided to act according to the following plan.

Each month he chooses several (possibly zero) of m buildings to demolish in such a way that renovation cost of **each of them separately** is not greater than the money tranche a_i of this month ($b_j \le a_i$) — it will allow to deceive city-dwellers that exactly this building will be renovated.

Then the mayor has to demolish all selected buildings during the current month as otherwise the dwellers will realize the deception and the plan will fail. Definitely the total demolishing cost can not exceed amount of money the mayor currently has. The mayor is not obliged to spend all the money on demolishing. If some money is left, the mayor puts it to the bank account and can use it in any subsequent month. Moreover, at any month he may choose not to demolish any buildings at all (in this case all the tranche will remain untouched and will be saved in the bank).

Your task is to calculate the maximal number of buildings the mayor can demolish.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 100\,000$) — the number of months before the championship and the number of ancient buildings in the city S.

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$), where a_i is the tranche of the i-th month.

The third line contains m integers $b_1, b_2, ..., b_m$ ($1 \le b_j \le 10^9$), where b_j is renovation cost of the j-th building.

The fourth line contains m integers $p_1, p_2, ..., p_m$ ($1 \le p_j \le 10^9$), where p_j is the demolishing cost of the j-th building.

Output

Output single integer — the maximal number of buildings the mayor can demolish.

Examples

input
2 3 2 4 6 2 3 1 3 2

output
2

input
3 5 5 3 1 5 2 9 1 10 4 2 1 3 10

output
3

input
5 6 6 3 2 4 3 3 6 4 5 4 2 1 4 3 2 5 3

output
6

Note

In the third example the mayor acts as follows.

In the first month he obtains 6 burles tranche and demolishes buildings #2 (renovation cost 6, demolishing cost 4) and #4 (renovation cost 5, demolishing cost 2). He spends all the money on it.

After getting the second month tranche of 3 burles, the mayor selects only building #1 (renovation cost 3, demolishing cost 1) for demolishing. As a result, he saves 2 burles for the next months.

In the third month he gets 2 burle tranche, but decides not to demolish any buildings at all. As a result, he has $2 + 2 = 4$ burles in the bank.

This reserve will be spent on the fourth month together with the 4-th tranche for demolishing of houses #3 and #5 (renovation cost is 4 for each, demolishing costs are 3 and 5 correspondingly). After this month his budget is empty.

Finally, after getting the last tranche of 3 burles, the mayor demolishes building #6 (renovation cost 2, demolishing cost 3).

As it can be seen, he demolished all 6 buildings.

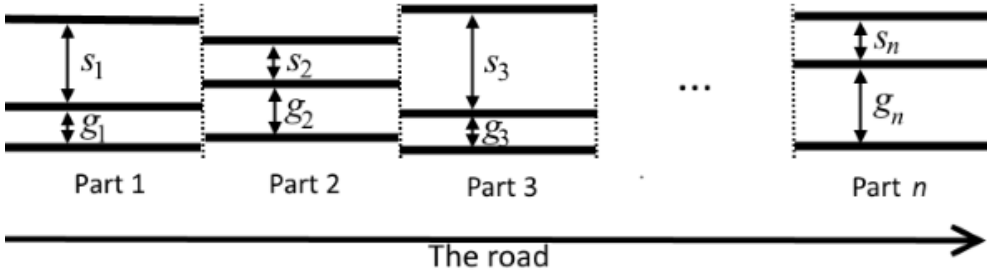
K. Road Widening

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Mayor of city S just hates trees and lawns. They take so much space and there could be a road on the place they occupy!

The Mayor thinks that one of the main city streets could be considerably widened on account of lawn nobody needs anyway. Moreover, that might help reduce the car jams which happen from time to time on the street.

The street is split into n equal length parts from left to right, the i-th part is characterized by two integers: width of road s_i and width of lawn g_i.



For each of n parts the Mayor should decide the size of lawn to demolish. For the i-th part he can reduce lawn width by integer x_i ($0 \le x_i \le g_i$). After it new road width of the i-th part will be equal to $s'_i = s_i + x_i$ and new lawn width will be equal to $g'_i = g_i - x_i$.

On the one hand, the Mayor wants to demolish as much lawn as possible (and replace it with road). On the other hand, he does not want to create a rapid widening or narrowing of the road, which would lead to car accidents. To avoid that, the Mayor decided that width of the road for consecutive parts should differ by at most 1, i.e. for each i ($1 \le i < n$) the inequation $|s'_{i+1} - s'_i| \le 1$ should hold. Initially this condition might not be true.

You need to find the the total width of lawns the Mayor will destroy according to his plan.

Input

The first line contains integer n ($1 \le n \le 2 \cdot 10^5$) — number of parts of the street.

Each of the following n lines contains two integers s_i, g_i ($1 \le s_i \le 10^6$, $0 \le g_i \le 10^6$) — current width of road and width of the lawn on the i-th part of the street.

Output

In the first line print the total width of lawns which will be removed.

In the second line print n integers $s'_1, s'_2, \ldots, s'_n$ ($s_i \le s'_i \le s_i + g_i$) — new widths of the road starting from the first part and to the last.

If there is no solution, print the only integer -1 in the first line.

Examples

input

```
3
4 5
4 5
4 10
```

output

```
16
9 9 10
```

input

```
4
1 100
100 1
1 100
100 1
```

output

```
202
101 101 101 101
```

input

```
3
1 1
100 100
1 1
```

output

```
-1
```

L. Berland.Taxi

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Berland.Taxi is a new taxi company with k cars which started operating in the capital of Berland just recently. The capital has n houses on a straight line numbered from 1 (leftmost) to n (rightmost), and the distance between any two neighboring houses is the same.

You have to help the company schedule all the taxi rides which come throughout the day according to the following rules:

- All cars are available for picking up passengers. Initially the j-th car is located next to the house with the number x_j at time 0.
- All cars have the same speed. It takes exactly 1 minute for any car to travel between neighboring houses i and $i + 1$.
- The i-th request for taxi ride comes at the time t_i, asking for a passenger to be picked up at the house a_i and dropped off at the house b_i. All requests for taxi rides are given in the increasing order of t_i. All t_i are distinct.

When a request for taxi ride is received at time t_i, Berland.Taxi operator assigns a car to it as follows:

- Out of cars which are currently available, operator assigns the car which is the *closest* to the pick up spot a_i. Needless to say, if a car is already on a ride with a passenger, it won't be available for any rides until that passenger is dropped off at the corresponding destination.
- If there are several such cars, operator will pick one of them which *has been waiting the most* since it became available.
- If there are several such cars, operator will pick one of them which *has the lowest number*.

After a car gets assigned to the taxi ride request:

- The driver immediately starts driving from current position to the house a_i.
- Once the car reaches house a_i, the passenger is immediately picked up and the driver starts driving to house b_i.
- Once house b_i is reached, the passenger gets dropped off and the car becomes available for new rides staying next to the house b_i.
- It is allowed for multiple cars to be located next to the same house at the same point in time, while waiting for ride requests or just passing by.

If there are no available cars at time t_i when a request for taxi ride comes, then:

- The i-th passenger will have to wait for a car to become available.
- When a car becomes available, operator will immediately assign it to this taxi ride request.
- If multiple cars become available at once while the passenger is waiting, operator will pick a car out of them according to the rules described above.

Operator processes taxi ride requests one by one. So if multiple passengers are waiting for the cars to become available, operator will not move on to processing the $(i + 1)$-th ride request until the car gets assigned to the i-th ride request.

Your task is to write a program that will process the given list of m taxi ride requests. For each request you have to find out which car will get assigned to it, and how long the passenger will have to wait for a car to arrive. Note, if there is already car located at the house a_i, then the corresponding wait time will be 0.

Input

The first line of input contains integers n, k and m ($2 \le n \le 2 \cdot 10^5$, $1 \le k, m \le 2 \cdot 10^5$) — number of houses, number of cars, and number of taxi ride requests. The second line contains integers $x_1, x_2, \ldots, x_k$ ($1 \le x_i \le n$) — initial positions of cars. x_i is a house number at which the i-th car is located initially. It's allowed for more than one car to be located next to the same house.

The following m lines contain information about ride requests. Each ride request is represented by integers t_j, a_j and b_j ($1 \le t_j \le 10^{12}$, $1 \le a_j, b_j \le n$, $a_j \ne b_j$), where t_j is time in minutes when a request is made, a_j is a house where passenger needs to be picked up, and b_j is a house where passenger needs to be dropped off. All taxi ride requests are given in the increasing order of t_j. All t_j are distinct.

Output

Print m lines: the j-th line should contain two integer numbers, the answer for the j-th ride request — *car number* assigned by the operator and *passenger wait time*.

Examples

input

```
10 1 2
3
5 2 8
9 10 3
```

output

```
1 1
1 5
```

input

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

output

```
1 2
```

input

```
5 2 2
1 5
10 3 5
20 4 1
```

output

```
1 2
2 1
```

Note

In the first sample test, a request comes in at time 5 and the car needs to get from house 3 to house 2 to pick up the passenger. Therefore wait time will be 1 and the ride will be completed at time $5 + 1 + 6 = 12$. The second request comes in at time 9, so the passenger will have to wait for the car to become available at time 12, and then the car needs another 2 minutes to get from house 8 to house 10. So the total wait time is $3 + 2 = 5$.

In the second sample test, cars 1 and 2 are located at the same distance from the first passenger and have the same "wait time since it became available". Car 1 wins a tiebreaker according to the rules because it has the lowest number. It will come to house 3 at time 3, so the wait time will be 2.

M. Quadcopter Competition

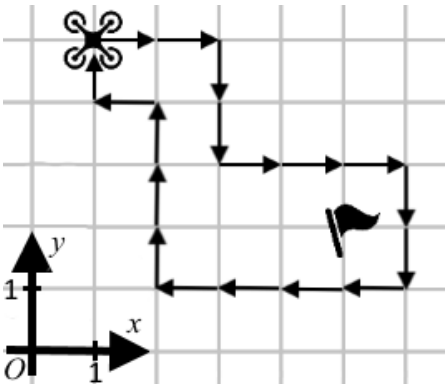
time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp takes part in a quadcopter competition. According to the rules a flying robot should:

- start the race from some point of a field,
- go around the flag,
- close cycle returning back to the starting point.

Polycarp knows the coordinates of the starting point (x_1, y_1) and the coordinates of the point where the flag is situated (x_2, y_2). Polycarp's quadcopter can fly only parallel to the sides of the field each tick changing exactly one coordinate by 1. It means that in one tick the quadcopter can fly from the point (x, y) to any of four points: $(x - 1, y)$, $(x + 1, y)$, $(x, y - 1)$ or $(x, y + 1)$.

Thus the quadcopter path is a closed cycle starting and finishing in (x_1, y_1) and containing the point (x_2, y_2) strictly inside.



The picture corresponds to the first example: the starting (and finishing) point is in $(1, 5)$ and the flag is in $(5, 2)$.

What is the minimal length of the quadcopter path?

Input

The first line contains two integer numbers x_1 and y_1 ($-100 \le x_1, y_1 \le 100$) — coordinates of the quadcopter starting (and finishing) point.

The second line contains two integer numbers x_2 and y_2 ($-100 \le x_2, y_2 \le 100$) — coordinates of the flag.

It is guaranteed that the quadcopter starting point and the flag do not coincide.

Output

Print the length of minimal path of the quadcopter to surround the flag and return back.

Examples

input
1 5 5 2
output
18

input
0 1 0 0
output
8