A. Phone Code

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

Polycarpus has n friends in Tarasov city. Polycarpus knows phone numbers of all his friends: they are strings $s_1, s_2, ..., s_n$. All these strings consist only of digits and have the same length.

Once Polycarpus needed to figure out Tarasov city phone code. He assumed that the phone code of the city is the longest common prefix of all phone numbers of his friends. In other words, it is the longest string c which is a prefix (the beginning) of each s_i for all i $(1 \le i \le n)$. Help Polycarpus determine the length of the city phone code.

Input

The first line of the input contains an integer n $(2 \le n \le 3 \cdot 10^4)$ — the number of Polycarpus's friends. The following n lines contain strings $s_1, s_2, ..., s_n$ — the phone numbers of Polycarpus's friends. It is guaranteed that all strings consist only of digits and have the same length from 1 to 20, inclusive. It is also guaranteed that all strings are different.

Output

Print the number of digits in the city phone code.

Sample test(s)

input

```
4
00209
00219
00999
00909
```

output

```
2
```

input

```
2
1
2
```

output

```
0
```

input

```
3
77012345678999999999
77012345678901234567
77012345678998765432
```

output

```
12
```

Note

A *prefix* of string t is a string that is obtained by deleting zero or more digits from the end of string t. For example, string "`00209`" has 6 prefixes: "" (an empty prefix), "`0`", "`00`", "`002`", "`0020`", "`00209`".

In the first sample the city phone code is string "`00`".

In the second sample the city phone code is an empty string.

In the third sample the city phone code is string "`770123456789`".

B. Pseudorandom Sequence Period

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

Polycarpus has recently got interested in sequences of pseudorandom numbers. He learned that many programming languages generate such sequences in a similar way: $r_i = (a \cdot r_{i-1} + b) \mod m$ (for $i \geq 1$). Here a, b, m are constants, fixed for the given realization of the pseudorandom numbers generator, r_0 is the so-called $randseed$ (this value can be set from the program using functions like `RandSeed(r)` or `srand(n)`), and mod denotes the operation of taking the remainder of division.

For example, if $a = 2, b = 6, m = 12, r_0 = 11$, the generated sequence will be: $4, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, \ldots$.

Polycarpus realized that any such sequence will sooner or later form a cycle, but the cycle may occur not in the beginning, so there exist a preperiod and a period. The example above shows a preperiod equal to 1 and a period equal to 2.

Your task is to find the period of a sequence defined by the given values of a, b, m and r_0. Formally, you have to find such minimum positive integer t, for which exists such positive integer k, that for any $i \geq k$: $r_i = r_{i+t}$.

Input

The single line of the input contains four integers a, b, m and r_0 ($1 \leq m \leq 10^5, 0 \leq a, b \leq 1000, 0 \leq r_0 < m$), separated by single spaces.

Output

Print a single integer — the period of the sequence.

Sample test(s)

input
```
2 6 12 11
```
output
```
2
```

input
```
2 3 5 1
```
output
```
4
```

input
```
3 6 81 9
```
output
```
1
```

Note

The first sample is described above.

In the second sample the sequence is (starting from the first element): $0, 3, 4, 1, 0, 3, 4, 1, 0, \ldots$

In the third sample the sequence is (starting from the first element): $33, 24, 78, 78, 78, 78, \ldots$

C. Bus

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

There is a bus stop near the university. The lessons are over, and n students come to the stop. The i-th student will appear at the bus stop at time t_i (all t_i's are distinct).

We shall assume that the stop is located on the coordinate axis Ox, at point $x = 0$, and the bus goes along the ray Ox, that is, towards the positive direction of the coordinate axis, and back. The i-th student needs to get to the point with coordinate x_i ($x_i > 0$).

The bus moves by the following algorithm. Initially it is at point 0. The students consistently come to the stop and get on it. The bus has a seating capacity which is equal to m passengers. At the moment when m students get on the bus, it starts moving in the positive direction of the coordinate axis. Also it starts moving when the last (n-th) student gets on the bus. The bus is moving at a speed of 1 unit of distance per 1 unit of time, i.e. it covers distance y in time y.

Every time the bus passes the point at which at least one student needs to get off, it stops and these students get off the bus. The students need $1 + [k / 2]$ units of time to get off the bus, where k is the number of students who leave at this point. Expression $[k / 2]$ denotes rounded down $k / 2$. As soon as the last student leaves the bus, the bus turns around and goes back to the point $x = 0$. It doesn't make any stops until it reaches the point. At the given point the bus fills with students once more, and everything is repeated.

If students come to the stop when there's no bus, they form a line (queue) and get on the bus in the order in which they came. Any number of students get on the bus in negligible time, you should assume that it doesn't take any time. Any other actions also take no time. The bus has no other passengers apart from the students.

Write a program that will determine for each student the time when he got off the bus. The moment a student got off the bus is the moment the bus stopped at the student's destination stop (despite the fact that the group of students need some time to get off).

Input

The first line contains two space-separated integers n, m ($1 \le n, m \le 10^5$) — the number of students and the number of passengers the bus can transport, correspondingly. Next n lines contain descriptions of the students, one per line. Each line contains a pair of integers t_i, x_i ($1 \le t_i \le 10^5$, $1 \le x_i \le 10^4$). The lines are given in the order of strict increasing of t_i. Values of x_i can coincide.

Output

Print n numbers $w_1, w_2, \ldots, w_n$, w_i — the moment of time when the i-th student got off the bus. Print the numbers on one line and separate them with single spaces.

Sample test(s)

input

```
1 10
3 5
```

output

```
8
```

input

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

output

```
8 19
```

input

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

output

```
11 11 11 11 20
```

input

```
20 4
28 13
31 13
35 6
36 4
52 6
53 4
83 2
84 4
```

```
87 1
93 6
108 4
113 6
116 1
125 2
130 2
136 13
162 2
166 4
184 1
192 2
```

output

```
51 51 43 40 93 89 86 89 114 121 118 121 137 139 139 152 195 199 193 195
```

Note

In the first sample the bus waits for the first student for 3 units of time and drives him to his destination in additional 5 units of time. So the student leaves the bus at the moment of time $3 + 5 = 8$.

In the second sample the capacity of the bus equals 1, that's why it will drive the first student alone. This student is the same as the student from the first sample. So the bus arrives to his destination at the moment of time 8, spends $1 + [1 / 2] = 1$ units of time on getting him off, and returns back to 0 in additional 5 units of time. That is, the bus returns to the bus stop at the moment of time 14. By this moment the second student has already came to the bus stop. So he immediately gets in the bus, and is driven to his destination in additional 5 units of time. He gets there at the moment $14 + 5 = 19$.

In the third sample the bus waits for the fourth student for 6 units of time, then drives for 5 units of time, then gets the passengers off for $1 + [4 / 2] = 3$ units of time, then returns for 5 units of time, and then drives the fifth student for 1 unit of time.

D. Calendar Reform

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

Reforms have started in Berland again! At this time, the Parliament is discussing the reform of the calendar. To make the lives of citizens of Berland more varied, it was decided to change the calendar. As more and more people are complaining that "the years fly by...", it was decided that starting from the next year the number of days per year will begin to grow. So the coming year will have exactly a days, the next after coming year will have $a + 1$ days, the next one will have $a + 2$ days and so on. This schedule is planned for the coming n years (in the n-th year the length of the year will be equal $a + n - 1$ day).

No one has yet decided what will become of months. An MP Palevny made the following proposal.

- The calendar for each month is comfortable to be printed on a square sheet of paper. We are proposed to make the number of days in each month be the square of some integer. The number of days per month should be the same for each month of any year, but may be different for different years.
- The number of days in each year must be divisible by the number of days per month in this year. This rule ensures that the number of months in each year is an integer.
- The number of days per month for each year must be chosen so as to save the maximum amount of paper to print the calendars. In other words, the number of days per month should be as much as possible.

These rules provide an unambiguous method for choosing the number of days in each month for any given year length. For example, according to Palevny's proposition, a year that consists of 108 days will have three months, 36 days each. The year that consists of 99 days will have 11 months, 9 days each, and a year of 365 days will have 365 months, one day each.

The proposal provoked heated discussion in the community, the famous mathematician Perelmanov quickly calculated that if the proposal is supported, then in a period of n years, beginning with the year that has a days, the country will spend p sheets of paper to print a set of calendars for these years. Perelmanov's calculations take into account the fact that the set will contain one calendar for each year and each month will be printed on a separate sheet.

Repeat Perelmanov's achievement and print the required number p. You are given positive integers a and n. Perelmanov warns you that your program should not work longer than four seconds at the maximum test.

Input

The only input line contains a pair of integers a, n ($1 \le a, n \le 10^7$; $a + n - 1 \le 10^7$).

Output

Print the required number p.

Please, do not use the %lld specifier to read or write 64-bit integers in C++. It is preferred to use cin, cout streams or the %I64d specifier.

Sample test(s)

input

```
25 3
```

output

```
30
```

input

```
50 5
```

output

```
125
```

Note

A note to the first sample test. A year of 25 days will consist of one month containing 25 days. A year of 26 days will consist of 26 months, one day each. A year of 27 days will have three months, 9 days each.

E. BHTML+BCSS

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is about imaginary languages BHTML and BCSS, which slightly resemble HTML and CSS. Read the problem statement carefully as the resemblance is rather slight and the problem uses very simplified analogs.

You are given a BHTML document that resembles HTML but is much simpler. It is recorded as a sequence of opening and closing tags. A tag that looks like "`<tagname>`" is called an opening tag and a tag that looks like "`</tagname>`" is called a closing tag. Besides, there are self-closing tags that are written as "`<tagname/>`" and in this problem they are fully equivalent to "`<tagname></tagname>`". All tagnames in this problem are strings consisting of lowercase Latin letters with length from 1 to 10 characters. Tagnames of different tags may coincide.

The document tags form a correct bracket sequence, that is, we can obtain an empty sequence from the given one using the following operations:

- remove any self-closing tag "`<tagname/>`",
- remove a pair of an opening and a closing tag that go consecutively (in this order) and have the same names. In other words, remove substring "`<tagname></tagname>`".

For example, you may be given such document: "`<header><p><a/><b></b></p></header><footer></footer>`" but you may not be given documents "`<a>`", "`<a></b>`", "`</a><a>`" or "`<a><b></a></b>`".

Obviously, for any opening tag there is the only matching closing one — each such pair is called an *element*. A self-closing tag also is an element. Let's consider that one element is nested inside another one, if tags of the first element are between tags of the second one. An element is not nested to itself. For instance, in the example above element "`b`" is nested in "`header`" and in "`p`", but it isn't nested in "`a`" and "`footer`", also it isn't nested to itself ("`b`"). Element "`header`" has three elements nested in it, and "`footer`" has zero.

We need the BCSS rules to apply styles when displaying elements of the BHTML documents. Each rule is recorded as a subsequence of words "x_1 x_2 ... x_n". This rule has effect over all such elements t, which satisfy both conditions from the list:

- there is a sequence of nested elements with tagnames "x_1", "x_2", ..., "x_n" (that is, the second element is nested in the first one, the third element is nested in the second one and so on),
- this sequence ends with element t (i.e. tagname of element t equals "x_n").

For example, element "`b`" meets the conditions of the rule "`a b`" if for element "`b`" exists element "`a`" in which it is nested. Element "`c`" meets the conditions of the rule "`a b b c`", if three elements exist: "`a`", "`b`", "`b`", and in the chain "`a`"-"`b`"-"`b`"-"`c`" each following element is nested in the previous one.

Given a BHTML document and a set of BCSS rules, write a program that determines the number of elements that meet the conditions of each rule.

Input

The first line of the input contains a BHTML-document. The document has length from 4 to 10^6 characters. The document has a correct structure, doesn't contain spaces or any other unnecessary characters. Tagnames consist of lowercase Latin letters, their lengths are from 1 to 10 characters.

The second line contains an integer m $(1 \le m \le 200)$ — the number of queries. Then m lines contain the queries, one per line. Each query is a sequence $x_1, x_2, ..., x_n$, where x_i is the i-th element of the query, and n $(1 \le n \le 200)$ is the number of elements in the query. The elements are separated by single spaces. Each query doesn't begin with and doesn't end with a space. Each query element is a sequence of lowercase Latin letters with length from 1 to 10.

Output

Print m lines, the j-th line should contain the number of elements of the document that correspond to the j-th BCSS-rule. If there are no such elements at all, print on the line `0`.

Sample test(s)

```
input
```

```
<a><b><b></b></b></a><a><b></b><b><v/></b></a><b></b>
4
a
a b b
a b
b a
```

```
output
```

```
2
1
4
0
```

```
input
```

```
<b><aa/></b><aa><b/><b/></aa>
5
aa b
b
aa
```

```
b aa
a
```

output

```
2
3
2
1
0
```