Codeforces Round #675 (Div. 2)

A. Fence

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

Yura is tasked to build a closed fence in shape of an arbitrary non-degenerate simple quadrilateral. He's already got three straight fence segments with known lengths a, b, and c. Now he needs to find out some possible integer length d of the fourth straight fence segment so that he can build the fence using these four segments. In other words, the fence should have a quadrilateral shape with side lengths equal to a, b, c, and d. Help Yura, find any possible length of the fourth side.

A non-degenerate simple quadrilateral is such a quadrilateral that no three of its corners lie on the same line, and it does not cross itself.

Input

The first line contains a single integer t — the number of test cases ($1 \leq t \leq 1000$). The next t lines describe the test cases.

Each line contains three integers a, b, and c — the lengths of the three fence segments ($1 \leq a, b, c \leq 10^9$).

Output

For each test case print a single integer d — the length of the fourth fence segment that is suitable for building the fence. If there are multiple answers, print any. We can show that an answer always exists.

Example

input

```
2
1 2 3
12 34 56
```

output

```
4
42
```

Note

We can build a quadrilateral with sides 1, 2, 3, 4.

We can build a quadrilateral with sides 12, 34, 56, 42.

B. Nice Matrix

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

A matrix of size $n \times m$ is called nice, if all rows and columns of the matrix are palindromes. A sequence of integers $(a_1, a_2, \dots, a_k)$ is a palindrome, if for any integer i ($1 \leq i \leq k$) the equality $a_i = a_{k-i+1}$ holds.

Sasha owns a matrix a of size $n \times m$. In one operation he can increase or decrease any number in the matrix by one. Sasha wants to make the matrix nice. He is interested what is the minimum number of operations he needs.

Help him!

Input

The first line contains a single integer t — the number of test cases ($1 \leq t \leq 10$). The t tests follow.

The first line of each test contains two integers n and m ($1 \leq n, m \leq 100$) — the size of the matrix.

Each of the next n lines contains m integers $a_{i,j}$ ($0 \leq a_{i,j} \leq 10^9$) — the elements of the matrix.

Output

For each test output the smallest number of operations required to make the matrix nice.

Example

input

```
2
4 2
4 2
```

```
2 4
4 2
2 4
3 4
1 2 3 4
5 6 7 8
9 10 11 18
```

output

```
8
42
```

Note

In the first test case we can, for example, obtain the following nice matrix in 8 operations:

```
2 2
4 4
4 4
2 2
```

In the second test case we can, for example, obtain the following nice matrix in 42 operations:

```
5 6 6 5
6 6 6 6
5 6 6 5
```

C. Bargain

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

Sometimes it is not easy to come to an agreement in a bargain. Right now Sasha and Vova can't come to an agreement: Sasha names a price as high as possible, then Vova wants to remove as many digits from the price as possible. In more details, Sasha names some integer price n, Vova removes a non-empty substring of (consecutive) digits from the price, the remaining digits close the gap, and the resulting integer is the price.

For example, is Sasha names 1213121, Vova can remove the substring 1312, and the result is 121.

It is allowed for result to contain leading zeros. If Vova removes all digits, the price is considered to be 0.

Sasha wants to come up with some constraints so that Vova can't just remove all digits, but he needs some arguments supporting the constraints. To start with, he wants to compute the sum of all possible resulting prices after Vova's move.

Help Sasha to compute this sum. Since the answer can be very large, print it modulo $10^9 + 7$.

Input

The first and only line contains a single integer n ($1 \le n < 10^{10^5}$).

Output

In the only line print the required sum modulo $10^9 + 7$.

Examples

input

```
107
```

output

```
42
```

input

```
100500100500
```

output

```
428101984
```

Note

Consider the first example.

Vova can choose to remove 1, 0, 7, 10, 07, or 107. The results are 07, 17, 10, 7, 1, 0. Their sum is 42.

D. Returning Home

time limit per test: 2 seconds

memory limit per test: 256 megabytes
input: standard input
output: standard output

Yura has been walking for some time already and is planning to return home. He needs to get home as fast as possible. To do this, Yura can use the instant-movement locations around the city.

Let's represent the city as an area of $n \times n$ square blocks. Yura needs to move from the block with coordinates (s_x, s_y) to the block with coordinates (f_x, f_y). In one minute Yura can move to any neighboring by side block; in other words, he can move in four directions. Also, there are m instant-movement locations in the city. Their coordinates are known to you and Yura. Yura can move to an instant-movement location in no time if he is located in a block with the same coordinate x or with the same coordinate y as the location.

Help Yura to find the smallest time needed to get home.

Input

The first line contains two integers n and m — the size of the city and the number of instant-movement locations ($1 \le n \le 10^9$, $0 \le m \le 10^5$).

The next line contains four integers s_x s_y f_x f_y — the coordinates of Yura's initial position and the coordinates of his home ($1 \le s_x, s_y, f_x, f_y \le n$).

Each of the next m lines contains two integers x_i y_i — coordinates of the i-th instant-movement location ($1 \le x_i, y_i \le n$).

Output

In the only line print the minimum time required to get home.

Examples

input

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

output

```
5
```

input

```
84 5
67 59 41 2
39 56
7 2
15 3
74 18
22 7
```

output

```
42
```

Note

In the first example Yura needs to reach $(5, 5)$ from $(1, 1)$. He can do that in 5 minutes by first using the second instant-movement location (because its y coordinate is equal to Yura's y coordinate), and then walking $(4, 1) \to (4, 2) \to (4, 3) \to (5, 3) \to (5, 4) \to (5, 5)$.

E. Minlexes

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

Some time ago Lesha found an entertaining string s consisting of lowercase English letters. Lesha immediately developed an unique algorithm for this string and shared it with you. The algorithm is as follows.

Lesha chooses an arbitrary (possibly zero) number of pairs on positions $(i, i+1)$ in such a way that the following conditions are satisfied:

- for each pair $(i, i+1)$ the inequality $0 \le i < |s| - 1$ holds;
- for each pair $(i, i+1)$ the equality $s_i = s_{i+1}$ holds;
- there is no index that is contained in more than one pair.

After that Lesha removes all characters on indexes contained in these pairs and the algorithm is over.
Lesha is interested in the lexicographically smallest strings he can obtain by applying the algorithm to the suffixes of the given string.

Input

The only line contains the string s ($1 \le |s| \le 10^5$) — the initial string consisting of lowercase English letters only.

Output

In $|s|$ lines print the lengths of the answers and the answers themselves, starting with the answer for the longest suffix. The output can be large, so, when some answer is longer than 10 characters, instead print the first 5 characters, then "...", then the last 2 characters of the answer.

Examples

input

```
abcdd
```

output

```
3 abc
2 bc
1 c
0
1 d
```

input

```
abbcdddeaaffdfouurtytwoo
```

output

```
18 abbcd...tw
17 bbcdd...tw
16 bcddd...tw
15 cddde...tw
14 dddea...tw
13 ddeaa...tw
12 deaad...tw
11 eaadf...tw
10 aadfortytw
9 adfortytw
8 dfortytw
9 fdfortytw
8 dfortytw
7 fortytw
6 ortytw
5 rtytw
6 urtytw
5 rtytw
4 tytw
3 ytw
2 tw
1 w
0
1 o
```

Note

Consider the first example.

- The longest suffix is the whole string "abcdd". Choosing one pair $(4, 5)$, Lesha obtains "abc".
- The next longest suffix is "bcdd". Choosing one pair $(3, 4)$, we obtain "bc".
- The next longest suffix is "cdd". Choosing one pair $(2, 3)$, we obtain "c".
- The next longest suffix is "dd". Choosing one pair $(1, 2)$, we obtain "" (an empty string).
- The last suffix is the string "d". No pair can be chosen, so the answer is "d".

In the second example, for the longest suffix "abbcdddeaaffdfouurtytwoo" choose three pairs $(11, 12)$, $(16, 17)$, $(23, 24)$ and we obtain "abbcdddeaadfortytw"

F. Boring Queries

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Yura owns a quite ordinary and boring array a of length n. You think there is nothing more boring than that, but Vladik doesn't agree!

In order to make Yura's array even more boring, Vladik makes q boring queries. Each query consists of two integers x and y. Before answering a query, the bounds l and r for this query are calculated: $l = (last + x) \bmod n + 1$, $r = (last + y) \bmod n + 1$, where $last$ is the answer on the previous query (zero initially), and $\bmod$ is the remainder operation. Whenever $l > r$, they are swapped.

After Vladik computes l and r for a query, he is to compute the least common multiple (LCM) on the segment $[l; r]$ of the initial array a modulo $10^9 + 7$. LCM of a multiset of integers is the smallest positive integer that is divisible by all the elements of the multiset. The obtained LCM is the answer for this query.

Help Vladik and compute the answer for each query!

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the length of the array.

The second line contains n integers a_i ($1 \le a_i \le 2 \cdot 10^5$) — the elements of the array.

The third line contains a single integer q ($1 \le q \le 10^5$) — the number of queries.

The next q lines contain two integers x and y each ($1 \le x, y \le n$) — the description of the corresponding query.

Output

Print q integers — the answers for the queries.

Example

input

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

output

```
6
2
15
30
```

Note

Consider the example:

- boundaries for first query are $(0+1) \bmod 3 + 1 = 2$ and $(0+3) \bmod 3 + 1 = 1$. LCM for segment $[1, 2]$ is equal to 6;
- boundaries for second query are $(6+3) \bmod 3 + 1 = 1$ and $(6+3) \bmod 3 + 1 = 1$. LCM for segment $[1, 1]$ is equal to 2;
- boundaries for third query are $(2+2) \bmod 3 + 1 = 2$ and $(2+3) \bmod 3 + 1 = 3$. LCM for segment $[2, 3]$ is equal to 15;
- boundaries for fourth query are $(15+2) \bmod 3 + 1 = 3$ and $(15+3) \bmod 3 + 1 = 1$. LCM for segment $[1, 3]$ is equal to 30.