AIM Tech Poorly Prepared Contest (unrated, funny, Div. 1 preferred)

A. Nash equilibrium

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

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

You are given a table A of integers $n \times m$. The cell (x, y) is called *Nash equilibrium* if both of the following conditions hold:

- for each $x_1 \neq x$ $A_{xy} > A_{x_1 y}$;
- for each $y_1 \neq y$ $A_{xy} < A_{xy_1}$.

Find a Nash equilibrium in A. If there exist several equilibria, print the one with minimum x. If still there are several possible answers, print the one with minimum y.

Input
The first line contains two integers n, m ($2 \leq n, m \leq 1000$), denoting the size of the table.

Each of next n lines contain m space-separated integers A_{ij} ($1 \leq A_{ij} \leq 10^9$).

Output
Output x and y — the coordinates of the lexicographically minimum Nash equilibrium in the table. In case there is no answer, print two zeroes instead.

Examples

input
4 4 1 2 3 4 1 2 3 5 1 2 3 6 2 3 5 7
output
1 4

input
3 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
output
0 0

B. DAG

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

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

You are given a directed acyclic graph G with n vertices and m edges. Denote by $R(v)$ the set of all vertices u reachable from v by moving along the edges of G. Find $\sum_{v=1}^{n} |R(v)|^2$.

Input
The first line contains two integers n, m ($1 \leq n, m \leq 5 \cdot 10^4$) denoting the number of vertices and the number of edges of G.

Each of the next m lines contains two integers u, v ($1 \leq u \neq v \leq n$), denoting the edge from u to v.

It's guaranteed that the given graph does not contain any cycles.

Output
Print one integer — the answer to the problem.

Examples

input
5 4 1 2 2 3 3 4 4 5
output
55

input
12 6 1 2 3 4 5 6 8 7 10 9 12 11
output
30

input
7 6 1 2 1 3 2 4 2 5 3 6 3 7
output
45

C. Segment tree or Fenwick?

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

You are given an array A of length n, initially filled with zeros. You need to process q queries to the array, each of one of the following types:

1. `1 x y`: you need to assign $A_x = y$;
2. `2 l r`: you need to print $\sum_{i=l}^{r} A_i$.

Furthermore, there are T independent tests you need to process.

Input

The first line contains an integer T ($1 \le T \le 10^5$) — the number of test cases.

Each test case description starts with two integers n, q ($1 \le n, q \le 10^5$) — the length of the array and the number of queries. The following q lines contain the description of queries: $1\ x\ y$ ($1 \le x \le n$, $0 \le y \le 10^9$) for queries of the first type and $2\ l\ r$ ($1 \le l \le r \le n$) for queries of the second type.

It is guaranteed that the sum of n as well as the sum of q does not exceed 10^6.

Output

For each query of the second type print its result on a separate line.

Example

input
2 6 5 2 1 6 1 3 2 2 2 4 1 6 3 2 1 6 5 3 1 3 7 1 1 4 2 1 5
output

```
0
2
5
11
```

D. Dijkstra

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

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

Find the distance between vertices 1 and n in an undirected weighted graph.

Input
The first line contains two integers n, m ($1 \le n, m \le 2 \cdot 10^5$) — the number of vertices and the number of edges.

Each of the next m lines contain description of an edge a_i b_i $cost_i$ ($1 \le a_i, b_i \le n$, $1 \le cost_i \le 10^9$).

Loops and multiple edges? Hmm, why not?

Output
Print one integer — the answer to the problem. If there is no path, print -1 instead.

Example

input
3 3 1 2 5 2 3 1 1 3 7
output
6

E. Amazing bitset

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

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

Andrey has just started to study competitive programming and he is fascinated by bitsets and operations on them.

For example, assume he has a bitset with n elements. He can change the values of the bitset in the following way:

1. He chooses different indices i_0, i_1, i_2, ... i_k ($k \ge 1$, $1 \le i_j \le n$) so that bit i_0 is different from all bits i_1, i_2, ... i_k.
2. He flips all these $k + 1$ bits.

He calls a bitset amazing if he can flip all the bits using several such changes.

Another topic Andrey is interested in is probability theory. For given n and p, where p is a rational number represented as $\frac{a}{b}$ with integer a, b, he considers all bitsets of length n, where each element is equal to 1 with probability p independently of the other elements.

He wants to know the probability that a bitset generated by the described method is amazing.

It is guaranteed that the answer to this problem can be represented as a rational number $\frac{x}{y}$, where y is coprime with $1\,234\,567\,891$. You need to output such integer z that $x \equiv yz \pmod{1\,234\,567\,891}$ and $0 \le z < 1\,234\,567\,891$.

Input
The only line contains three integers n, a, b ($1 \le n \le 10^9$, $0 \le a \le 10^5$, $1 \le b \le 10^5$, $a \le b$), denoting the length of the bitset and the probability of an element to be 1 (probability is $\frac{a}{b}$).

Output
Output the answer to the problem in the only line.

Examples

input
5 1 2
output
848765426

input
1 228 239
output
0

F. Keep talking and nobody explodes – easy

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

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

You have the safe lock which consists of 5 decimal digits. If you rotate some digit, it increases by one, except 9 which becomes 0.

Initially, the lock contains number x. To unlock the safe you must do the following operations in order (and be careful, don't mix up if and else statements).

If sum of digits on positions 1 and 4 is greater than 10, rotate digit on position 1 by 3 times, else rotate digit on position 4 by 8 times.

If sum of digits on positions 3 and 2 is greater than 8, rotate digit on position 4 by 9 times, else rotate digit on position 5 by 8 times.

If digit on position 3 is odd, rotate digit on position 3 by 3 times, else rotate digit on position 3 by 4 times.

If digit on position 5 is greater than digit on position 2, rotate digit on position 4 by 1 times, else rotate digit on position 2 by 7 times.

If digit on position 1 is odd, rotate digit on position 1 by 3 times, else rotate digit on position 3 by 5 times.

If digit on position 4 is odd, rotate digit on position 4 by 7 times, else rotate digit on position 1 by 9 times.

If digit on position 4 is greater than digit on position 1, rotate digit on position 4 by 9 times, else rotate digit on position 4 by 2 times.

If digit on position 1 is greater than digit on position 3, rotate digit on position 2 by 1 times, else rotate digit on position 3 by 1 times.

If digit on position 5 is greater than digit on position 3, rotate digit on position 4 by 5 times, else rotate digit on position 5 by 8 times.

If sum of digits on positions 1 and 3 is greater than 8, rotate digit on position 4 by 5 times, else rotate digit on position 2 by 5 times.

If digit on position 1 is greater than digit on position 4, rotate digit on position 4 by 3 times, else rotate digit on position 2 by 3 times.

If sum of digits on positions 3 and 1 is greater than 9, rotate digit on position 2 by 9 times, else rotate digit on position 2 by 2 times.

If sum of digits on positions 4 and 3 is greater than 10, rotate digit on position 4 by 7 times, else rotate digit on position 5 by 7 times.

If digit on position 3 is greater than digit on position 2, rotate digit on position 3 by 2 times, else rotate digit on position 4 by 6 times.

If digit on position 1 is greater than digit on position 3, rotate digit on position 1 by 9 times, else rotate digit on position 2 by 9 times.

If digit on position 3 is odd, rotate digit on position 3 by 9 times, else rotate digit on position 1 by 5 times.

If sum of digits on positions 3 and 5 is greater than 9, rotate digit on position 3 by 4 times, else rotate digit on position 3 by 9 times.

If digit on position 3 is greater than digit on position 1, rotate digit on position 5 by 1 times, else rotate digit on position 5 by 7 times.

If digit on position 1 is greater than digit on position 3, rotate digit on position 2 by 9 times, else rotate digit on position 4 by 6 times.

If sum of digits on positions 2 and 3 is greater than 10, rotate digit on position 2 by 2 times, else rotate digit on position 3 by 6 times.

Input

Input contains single number x consisting of exactly 5 digits, leading zeroes are allowed.

Output

Output the number after applying all operations.

Examples

input
00000
output
61376

input
12345

output
07769

G. Keep talking and nobody explodes – medium

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

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

You have the safe lock which consists of 5 decimal digits. If you rotate some digit, it increases by one, except 9 which becomes 0.

Initially, the lock contains number x. To unlock the safe you must do the following operations in order (and be careful, don't mix up if and else statements).

If sum of digits on positions 3 and 4 is greater than 10, rotate digit on position 2 by 9 times, else rotate digit on position 5 by 4 times.

If digit on position 3 is greater than digit on position 5, rotate digit on position 2 by 4 times, else rotate digit on position 4 by 6 times.

If digit on position 5 is greater than digit on position 3, rotate digit on position 2 by 1 times, else rotate digit on position 1 by 7 times.

If sum of digits on positions 4 and 1 is greater than 8, rotate digit on position 2 by 7 times, else rotate digit on position 3 by 3 times.

If digit on position 4 is greater than digit on position 1, rotate digit on position 2 by 3 times, else rotate digit on position 3 by 2 times.

If sum of digits on positions 1 and 2 is greater than 9, rotate digit on position 3 by 6 times, else rotate digit on position 5 by 3 times.

If digit on position 3 is greater than digit on position 2, rotate digit on position 1 by 2 times, else rotate digit on position 4 by 5 times.

If digit on position 5 is greater than digit on position 3, rotate digit on position 2 by 1 times, else rotate digit on position 4 by 5 times.

If sum of digits on positions 5 and 1 is greater than 10, rotate digit on position 4 by 7 times, else rotate digit on position 3 by 5 times.

If sum of digits on positions 5 and 4 is greater than 9, rotate digit on position 3 by 9 times, else rotate digit on position 2 by 4 times.

If sum of digits on positions 3 and 1 is greater than 8, rotate digit on position 2 by 8 times, else rotate digit on position 4 by 4 times.

If digit on position 5 is greater than digit on position 2, rotate digit on position 1 by 5 times, else rotate digit on position 3 by 8 times.

If sum of digits on positions 1 and 4 is greater than 10, rotate digit on position 3 by 4 times, else rotate digit on position 5 by 1 times.

If digit on position 3 is greater than digit on position 5, rotate digit on position 2 by 1 times, else rotate digit on position 1 by 6 times.

If sum of digits on positions 1 and 5 is greater than 9, rotate digit on position 3 by 3 times, else rotate digit on position 2 by 1 times.

If digit on position 5 is greater than digit on position 1, rotate digit on position 4 by 8 times, else rotate digit on position 2 by 1 times.

If digit on position 4 is greater than digit on position 1, rotate digit on position 3 by 4 times, else rotate digit on position 5 by 4 times.

If sum of digits on positions 3 and 1 is greater than 8, rotate digit on position 5 by 3 times, else rotate digit on position 2 by 6 times.

If digit on position 3 is greater than digit on position 4, rotate digit on position 2 by 3 times, else rotate digit on position 1 by 5 times.

If digit on position 5 is greater than digit on position 4, rotate digit on position 2 by 7 times, else rotate digit on position 3 by 8 times.

If digit on position 2 is greater than digit on position 4, rotate digit on position 5 by 9 times, else rotate digit on position 1 by 4 times.

If sum of digits on positions 3 and 5 is greater than 10, rotate digit on position 4 by 1 times, else rotate digit on position 2 by 5 times.

If digit on position 4 is greater than digit on position 1, rotate digit on position 3 by 9 times, else rotate digit on position 2 by 9 times.

If digit on position 5 is greater than digit on position 3, rotate digit on position 2 by 4 times, else rotate digit on position 1 by 6 times.

If sum of digits on positions 3 and 4 is greater than 9, rotate digit on position 5 by 8 times, else rotate digit on position 2 by 5 times.

If sum of digits on positions 3 and 4 is greater than 10, rotate digit on position 5 by 2 times, else rotate digit on position 1 by 5 times.

If sum of digits on positions 5 and 4 is greater than 9, rotate digit on position 3 by 3 times, else rotate digit on position 1 by 8 times.

If digit on position 5 is greater than digit on position 2, rotate digit on position 1 by 4 times, else rotate digit on position 3 by 8 times.

If digit on position 3 is greater than digit on position 1, rotate digit on position 5 by 6 times, else rotate digit on position 2 by 6 times.

If digit on position 4 is greater than digit on position 5, rotate digit on position 1 by 6 times, else rotate digit on position 3 by 1 times.

If sum of digits on positions 3 and 5 is greater than 10, rotate digit on position 2 by 5 times, else rotate digit on position 1 by 7 times.

If sum of digits on positions 5 and 2 is greater than 9, rotate digit on position 4 by 9 times, else rotate digit on position 3 by 5 times.

If sum of digits on positions 2 and 4 is greater than 10, rotate digit on position 3 by 1 times, else rotate digit on position 1 by 2 times.

If digit on position 3 is greater than digit on position 4, rotate digit on position 5 by 7 times, else rotate digit on position 2 by 1 times.

If digit on position 2 is greater than digit on position 5, rotate digit on position 1 by 6 times, else rotate digit on position 4 by 2 times.

If digit on position 2 is greater than digit on position 1, rotate digit on position 5 by 3 times, else rotate digit on position 4 by 4 times.

If digit on position 5 is greater than digit on position 4, rotate digit on position 3 by 9 times, else rotate digit on position 1 by 9 times.

If digit on position 1 is greater than digit on position 5, rotate digit on position 4 by 6 times, else rotate digit on position 2 by 5 times.

If sum of digits on positions 1 and 5 is greater than 10, rotate digit on position 3 by 7 times, else rotate digit on position 2 by 4 times.

If sum of digits on positions 2 and 1 is greater than 9, rotate digit on position 3 by 7 times, else rotate digit on position 5 by 4 times.

Input
Input contains single number x consisting of exactly 5 digits, leading zeroes are allowed.

Output
Output the number after applying all operations.

Examples

input
00000
output
43266

input
12345
output
87229

H. Who needs suffix structures?

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

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

You run a string shop. During a day your customers want to buy strings of certain lengths and sometimes satisfying other properties.

You have a string of length n and can cut a string of any length starting at any position out of it. Today your first customer asks you some questions of type "Is there any difference between substrings of length k if one of them starts at position i and another one starts from the j-th position?" You are to answer all these questions.

Please note that in your shop strings are over an alphabet of size $987\,898\,789$.

Input
The first line contains two integers n and q ($1 \le q, n \le 200\,000$). The second line contains n space-separated integers a_1, ..., a_n ($0 \le a_i < 987\,898\,789$) which denote the letters of string s, and each of the following q lines contains three integers len, pos_1 and pos_2 ($1 \le len \le n$, $1 \le pos_1, pos_2 \le n - len + 1$) denoting the corresponding query.

Output
Print q lines, i-th of them containing Yes if the two substrings in the i-th query (of length len starting from pos_1 and pos_2) are equal and No otherwise.

Examples

input
5 2 1 2 3 1 2 2 1 4 3 1 3
output
Yes No

input
9 3 0 0 0 0 0 0 0 0 0 9 1 1 8 1 2 1 1 9

output
Yes Yes Yes

I. Deja vu

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

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

You run a string shop. During a day your customers want to buy strings of certain lengths and sometimes satisfying other properties.

Today your first customer asked you if you have a string of length k. In fact, you have a string of length n and can cut a string of length k starting at any position out of it. You wonder how many distinct values this string can equal.

Please note that in your shop strings are over an alphabet of size 2.

Input
The first line contains two integers n and k ($1 \leq k \leq n \leq 200\,000$). The second line contains a binary string s of length n.

Output
In the only line print the number of distinct substrings of length k of the string s.

Examples

input
5 2 01101

output
3

input
10 3 1110001011

output
8

Note
The second sample test represents the de Bruijn sequence of order 3.

J. Keep talking and nobody explodes – hard

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

This is an unusual problem in an unusual contest, here is the announcement: http://codeforces.com/blog/entry/73543

You have the safe lock which consists of 100 decimal digits. If you rotate some digit, it increases by one, except 9 which becomes 0.

Initially, the lock contains number x. To unlock the safe you must do the following operations in order (and be careful, don't mix up if and else statements).

If digit 39 is odd, rotate digit 39 by 9 times, else rotate digit 37 by 1 times. If digit 24 is odd, rotate digit 24 by 1 times, else rotate digit 76 by 3 times. If sum of digits 13 and 91 is greater than 10, rotate digit 14 by 6 times, else rotate digit 34 by 8 times. If digit 87 is odd, rotate digit 87 by 7 times, else rotate digit 22 by 9 times. If digit 79 is greater than digit 15, rotate digit 74 by 7 times, else rotate digit 84 by 6 times. If sum of digits 26 and 66 is greater than 9, rotate digit 31 by 7 times, else rotate digit 95 by 4 times. If sum of digits 53 and 1 is greater than 8, rotate digit 66 by 1 times, else rotate digit 94 by 6 times. If digit 41 is greater than digit 29, rotate digit 67 by 5 times, else rotate digit 41 by 9 times. If sum of digits 79 and 20 is greater than 10, rotate digit 18 by 2 times, else rotate digit 72 by 9 times. If sum of digits 14 and 24 is greater than 10, rotate digit 64 by 2 times, else rotate digit 84 by 2 times. If digit 16 is greater than digit 34, rotate digit 81 by 5 times, else rotate digit 15 by 2 times. If sum of digits 48 and 65 is greater than 9, rotate digit 57 by 2 times, else rotate digit 28 by 5 times. If digit 81 is odd, rotate digit 81 by 5 times, else rotate digit 25 by 4 times. If digit 70 is odd, rotate digit 70 by 9 times, else rotate digit 93 by 3 times. If sum of digits 92 and 49 is greater

than 9, rotate digit 81 by 2 times, else rotate digit 42 by 3 times. If digit 96 is greater than digit 20, rotate digit 45 by 4 times, else rotate digit 45 by 1 times. If digit 91 is greater than digit 21, rotate digit 60 by 3 times, else rotate digit 72 by 1 times. If digit 89 is greater than digit 7, rotate digit 98 by 9 times, else rotate digit 52 by 7 times. If digit 38 is greater than digit 97, rotate digit 92 by 6 times, else rotate digit 35 by 4 times. If digit 96 is greater than digit 99, rotate digit 42 by 4 times, else rotate digit 40 by 9 times. If digit 86 is odd, rotate digit 86 by 1 times, else rotate digit 14 by 3 times. If digit 23 is odd, rotate digit 23 by 5 times, else rotate digit 55 by 9 times. If digit 79 is odd, rotate digit 79 by 1 times, else rotate digit 29 by 8 times. If digit 4 is greater than digit 91, rotate digit 98 by 8 times, else rotate digit 69 by 4 times. If digit 93 is greater than digit 24, rotate digit 75 by 9 times, else rotate digit 95 by 3 times. If sum of digits 32 and 50 is greater than 10, rotate digit 91 by 3 times, else rotate digit 1 by 5 times. If digit 81 is greater than digit 31, rotate digit 86 by 7 times, else rotate digit 67 by 5 times. If digit 83 is greater than digit 86, rotate digit 48 by 7 times, else rotate digit 2 by 6 times. If digit 20 is greater than digit 88, rotate digit 9 by 2 times, else rotate digit 99 by 4 times. If digit 14 is odd, rotate digit 14 by 5 times, else rotate digit 97 by 7 times. If digit 38 is greater than digit 14, rotate digit 48 by 2 times, else rotate digit 81 by 5 times. If digit 92 is greater than digit 74, rotate digit 92 by 1 times, else rotate digit 50 by 9 times. If digit 76 is greater than digit 89, rotate digit 68 by 6 times, else rotate digit 69 by 5 times. If digit 2 is greater than digit 28, rotate digit 75 by 1 times, else rotate digit 89 by 1 times. If digit 67 is odd, rotate digit 67 by 9 times, else rotate digit 49 by 1 times. If digit 23 is odd, rotate digit 23 by 1 times, else rotate digit 59 by 3 times. If digit 81 is odd, rotate digit 81 by 9 times, else rotate digit 9 by 4 times. If sum of digits 92 and 82 is greater than 9, rotate digit 81 by 2 times, else rotate digit 91 by 5 times. If sum of digits 42 and 48 is greater than 9, rotate digit 35 by 8 times, else rotate digit 59 by 6 times. If digit 55 is odd, rotate digit 55 by 9 times, else rotate digit 61 by 6 times. If digit 83 is odd, rotate digit 83 by 5 times, else rotate digit 85 by 4 times. If digit 96 is odd, rotate digit 96 by 1 times, else rotate digit 72 by 4 times. If digit 17 is odd, rotate digit 17 by 1 times, else rotate digit 28 by 3 times. If digit 85 is greater than digit 74, rotate digit 37 by 3 times, else rotate digit 10 by 3 times. If sum of digits 50 and 67 is greater than 9, rotate digit 85 by 9 times, else rotate digit 42 by 4 times. If sum of digits 11 and 43 is greater than 10, rotate digit 56 by 7 times, else rotate digit 50 by 7 times. If sum of digits 95 and 64 is greater than 9, rotate digit 95 by 4 times, else rotate digit 95 by 9 times. If sum of digits 21 and 16 is greater than 9, rotate digit 87 by 3 times, else rotate digit 30 by 1 times. If digit 91 is odd, rotate digit 91 by 1 times, else rotate digit 77 by 1 times. If digit 95 is greater than digit 82, rotate digit 53 by 2 times, else rotate digit 100 by 5 times. If sum of digits 88 and 66 is greater than 10, rotate digit 34 by 4 times, else rotate digit 57 by 4 times. If digit 73 is greater than digit 84, rotate digit 52 by 3 times, else rotate digit 42 by 9 times. If digit 66 is greater than digit 38, rotate digit 94 by 7 times, else rotate digit 78 by 7 times. If digit 23 is greater than digit 12, rotate digit 78 by 2 times, else rotate digit 62 by 8 times. If digit 13 is greater than digit 9, rotate digit 42 by 7 times, else rotate digit 1 by 9 times. If digit 43 is greater than digit 29, rotate digit 20 by 2 times, else rotate digit 47 by 2 times. If sum of digits 100 and 51 is greater than 8, rotate digit 10 by 6 times, else rotate digit 89 by 1 times. If digit 19 is greater than digit 37, rotate digit 26 by 7 times, else rotate digit 30 by 8 times. If digit 73 is greater than digit 25, rotate digit 77 by 3 times, else rotate digit 41 by 1 times. If sum of digits 67 and 96 is greater than 10, rotate digit 47 by 6 times, else rotate digit 33 by 5 times. If digit 11 is greater than digit 10, rotate digit 33 by 3 times, else rotate digit 4 by 3 times. If digit 85 is odd, rotate digit 85 by 7 times, else rotate digit 37 by 9 times. If digit 14 is odd, rotate digit 14 by 1 times, else rotate digit 28 by 4 times. If sum of digits 30 and 18 is greater than 8, rotate digit 93 by 5 times, else rotate digit 68 by 1 times. If sum of digits 54 and 72 is greater than 8, rotate digit 88 by 8 times, else rotate digit 25 by 8 times. If digit 72 is odd, rotate digit 72 by 5 times, else rotate digit 10 by 3 times. If digit 15 is odd, rotate digit 15 by 3 times, else rotate digit 68 by 1 times. If sum of digits 81 and 31 is greater than 9, rotate digit 2 by 5 times, else rotate digit 35 by 1 times. If digit 57 is odd, rotate digit 57 by 1 times, else rotate digit 25 by 9 times. If sum of digits 75 and 51 is greater than 9, rotate digit 73 by 8 times, else rotate digit 49 by 1 times. If sum of digits 81 and 61 is greater than 10, rotate digit 61 by 3 times, else rotate digit 88 by 1 times. If digit 60 is odd, rotate digit 60 by 1 times, else rotate digit 31 by 2 times. If digit 93 is odd, rotate digit 93 by 5 times, else rotate digit 50 by 1 times. If sum of digits 19 and 82 is greater than 9, rotate digit 48 by 7 times, else rotate digit 88 by 8 times. If digit 45 is odd, rotate digit 45 by 7 times, else rotate digit 100 by 1 times. If digit 46 is greater than digit 71, rotate digit 28 by 8 times, else rotate digit 37 by 6 times. If digit 79 is odd, rotate digit 79 by 5 times, else rotate digit 10 by 1 times. If digit 19 is greater than digit 95, rotate digit 76 by 9 times, else rotate digit 95 by 8 times. If digit 49 is odd, rotate digit 49 by 5 times, else rotate digit 66 by 3 times. If digit 62 is odd, rotate digit 62 by 1 times, else rotate digit 26 by 8 times. If digit 67 is greater than digit 33, rotate digit 27 by 8 times, else rotate digit 96 by 2 times. If sum of digits 73 and 15 is greater than 8, rotate digit 98 by 6 times, else rotate digit 11 by 6 times. If digit 63 is greater than digit 42, rotate digit 66 by 1 times, else rotate digit 58 by 2 times. If digit 41 is odd, rotate digit 41 by 9 times, else rotate digit 99 by 5 times. If digit 93 is odd, rotate digit 93 by 5 times, else rotate digit 53 by 1 times. If digit 46 is odd, rotate digit 46 by 3 times, else rotate digit 64 by 4 times. If sum of digits 99 and 64 is greater than 10, rotate digit 72 by 9 times, else rotate digit 51 by 5 times. If digit 75 is greater than digit 23, rotate digit 89 by 2 times, else rotate digit 76 by 7 times. If digit 6 is odd, rotate digit 6 by 1 times, else rotate digit 44 by 6 times. If digit 58 is odd, rotate digit 58 by 3 times, else rotate digit 49 by 9 times. If digit 5 is greater than digit 13, rotate digit 46 by 9 times, else rotate digit 21 by 7 times. If sum of digits 44 and 94 is greater than 9, rotate digit 36 by 4 times, else rotate digit 15 by 3 times. If sum of digits 52 and 43 is greater than 8, rotate digit 29 by 8 times, else rotate digit 72 by 6 times. If sum of digits 87 and 48 is greater than 9, rotate digit 61 by 8 times, else rotate digit 14 by 3 times. If digit 81 is odd, rotate digit 81 by 7 times, else rotate digit 64 by 2 times. If digit 88 is odd, rotate digit 88 by 7 times, else rotate digit 53 by 9 times. If sum of digits 86 and 78 is greater than 10, rotate digit 96 by 7 times, else rotate digit 79 by 1 times. If digit 20 is odd, rotate digit 20 by 7 times, else rotate digit 2 by 7 times. If digit 77 is greater than digit 80, rotate digit 60 by 5 times, else rotate digit 38 by 8 times. If digit 65 is odd, rotate digit 65 by 1 times, else rotate digit 85 by 3 times.

Input

Input contains single number x consisting of exactly 100 digits, leading zeroes are allowed.

Output

Output the number after applying all operations.

Examples

input
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
output
97000100060003000200001117066170349430999700128610009087000931041457490807063260605070701046037276 96

input
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

output
043457783912373680908195967879121496389995349995506224434859433857759911345310600230237400428748413 6