A. Soft Drinking

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

This winter is so cold in Nvodsk! A group of n friends decided to buy k bottles of a soft drink called "Take-It-Light" to warm up a bit. Each bottle has l milliliters of the drink. Also they bought c limes and cut each of them into d slices. After that they found p grams of salt.

To make a toast, each friend needs nl milliliters of the drink, a slice of lime and np grams of salt. The friends want to make as many toasts as they can, provided they all drink the same amount. How many toasts can each friend make?

Input

The first and only line contains **positive** integers n, k, l, c, d, p, nl, np, not exceeding 1000 and no less than 1. The numbers are separated by exactly one space.

Output

Print a single integer — the number of toasts each friend can make.

Sample test(s)

input
```
3 4 5 10 8 100 3 1
```
output
```
2
```

input
```
5 100 10 1 19 90 4 3
```
output
```
3
```

input
```
10 1000 1000 25 23 1 50 1
```
output
```
0
```

Note

A comment to the first sample:

Overall the friends have $4 * 5 = 20$ milliliters of the drink, it is enough to make $20 / 3 = 6$ toasts. The limes are enough for $10 * 8 = 80$ toasts and the salt is enough for $100 / 1 = 100$ toasts. However, there are 3 friends in the group, so the answer is $min(6, 80, 100) / 3 = 2$.

B. Phone Numbers

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

Winters are just damn freezing cold in Nvodsk! That's why a group of n friends prefers to take a taxi, order a pizza and call girls. The phone numbers in the city consist of three pairs of digits (for example, 12-34-56). Each friend has a phonebook of size s_i (that's the number of phone numbers). We know that taxi numbers consist of six identical digits (for example, 22-22-22), the numbers of pizza deliveries should necessarily be decreasing sequences of six different digits (for example, 98-73-21), all other numbers are the girls' numbers.

You are given your friends' phone books. Calculate which friend is best to go to when you are interested in each of those things (who has maximal number of phone numbers of each type).

If the phone book of one person contains some number two times, you should count it **twice**. That is, each number should be taken into consideration the number of times it occurs in the phone book.

Input

The first line contains an integer n $(1 \le n \le 100)$ — the number of friends.

Then follow n data blocks that describe each friend's phone books. Each block is presented in the following form: first goes the line that contains integer s_i and string $name_i$ $(0 \le s_i \le 100)$ — the number of phone numbers in the phone book of the i-th friend and the name of the i-th friend. The name is a non-empty sequence of uppercase and lowercase Latin letters, containing no more than 20 characters. Next s_i lines contain numbers as "XX-XX-XX", where X is arbitrary digits from 0 to 9.

Output

In the first line print the phrase "`If you want to call a taxi, you should call: `". Then print names of all friends whose phone books contain maximal number of taxi phone numbers.

In the second line print the phrase "`If you want to order a pizza, you should call: `". Then print names of all friends who have maximal number of pizza phone numbers.

In the third line print the phrase "`If you want to go to a cafe with a wonderful girl, you should call: `". Then print names of all friends who have maximal number of girls' phone numbers.

Print the names in the order in which **they are given in the input data**. Separate two consecutive names with a comma and a space. Each line should end with exactly one point. For clarifications concerning the output form, see sample tests. It is necessary that you **follow the output form strictly**. Extra spaces are not allowed.

Sample test(s)

input

```
4
2 Fedorov
22-22-22
98-76-54
3 Melnikov
75-19-09
23-45-67
99-99-98
7 Rogulenko
22-22-22
11-11-11
33-33-33
44-44-44
55-55-55
66-66-66
95-43-21
3 Kaluzhin
11-11-11
99-99-99
98-65-32
```

output

```
If you want to call a taxi, you should call: Rogulenko.
If you want to order a pizza, you should call: Fedorov, Rogulenko, Kaluzhin.
If you want to go to a cafe with a wonderful girl, you should call: Melnikov.
```

input

```
3
5 Gleb
66-66-66
55-55-55
01-01-01
65-43-21
12-34-56
3 Serega
55-55-55
87-65-43
65-55-21
```

```
5 Melnik
12-42-12
87-73-01
36-04-12
88-12-22
82-11-43
```

output

```
If you want to call a taxi, you should call: Gleb.
If you want to order a pizza, you should call: Gleb, Serega.
If you want to go to a cafe with a wonderful girl, you should call: Melnik.
```

input

```
3
3 Kulczynski
22-22-22
65-43-21
98-12-00
4 Pachocki
11-11-11
11-11-11
11-11-11
98-76-54
0 Smietanka
```

output

```
If you want to call a taxi, you should call: Pachocki.
If you want to order a pizza, you should call: Kulczynski, Pachocki.
If you want to go to a cafe with a wonderful girl, you should call: Kulczynski.
```

Note

In the first sample you are given four friends. `Fedorov`'s phone book contains one taxi number and one pizza delivery number, `Melnikov`'s phone book only has 3 numbers of girls, `Rogulenko`'s one has 6 taxi numbers and one pizza delivery number, `Kaluzhin`'s one contains 2 taxi numbers and one pizza delivery number.

Thus, if you need to order a taxi, you should obviously call `Rogulenko`, if you need to order a pizza you should call anybody of the following: `Rogulenko`, `Fedorov`, `Kaluzhin` (each of them has one number). `Melnikov` has maximal number of phone numbers of girls.

C. Win or Freeze

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

You can't possibly imagine how cold our friends are this winter in Nvodsk! Two of them play the following game to warm up: initially a piece of paper has an integer q. During a move a player should write any integer number that is a *non-trivial* divisor of the last written number. Then he should run this number of circles around the hotel. Let us remind you that a number's divisor is called *non-trivial* if it is different from one and from the divided number itself.

The first person who **can't make a move wins** as he continues to lie in his warm bed under three blankets while the other one keeps running. Determine which player wins considering that both players play optimally. If the first player wins, print any winning first move.

Input

The first line contains the only integer q ($1 \le q \le 10^{13}$).

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

Output

In the first line print the number of the winning player (1 or 2). If the first player wins then the second line should contain another integer — his first move (if the first player can't even make the first move, print 0). If there are multiple solutions, print any of them.

Sample test(s)

input
```
6
```
output
```
2
```

input
```
30
```
output
```
1
6
```

input
```
1
```
output
```
1
0
```

Note

Number 6 has only two non-trivial divisors: 2 and 3. It is impossible to make a move after the numbers 2 and 3 are written, so both of them are winning, thus, number 6 is the losing number. A player can make a move and write number 6 after number 30; 6, as we know, is a losing number. Thus, this move will bring us the victory.

D. Quantity of Strings

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

Just in case somebody missed it: this winter is totally cold in Nvodsk! It is so cold that one gets funny thoughts. For example, let's say there are strings with the length exactly n, based on the alphabet of size m. Any its substring with length equal to k is a palindrome. How many such strings exist? Your task is to find their quantity modulo 1000000007 ($10^9 + 7$). Be careful and don't miss a string or two!

Let us remind you that a string is a palindrome if it can be read the same way in either direction, from the left to the right and from the right to the left.

Input

The first and only line contains three integers: n, m and k ($1 \le n, m, k \le 2000$).

Output

Print a single integer — the number of strings of the described type modulo 1000000007 ($10^9 + 7$).

Sample test(s)

input

```
1 1 1
```

output

```
1
```

input

```
5 2 4
```

output

```
2
```

Note

In the first sample only one string is valid: "a" (let's denote the only letter of our alphabet as "a").

In the second sample (if we denote the alphabet letters as "a" and "b") the following strings are valid: "aaaaa" and "bbbbb".

E. Smart Cheater

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

I guess there's not much point in reminding you that Nvodsk winters aren't exactly hot. That increased the popularity of the public transport dramatically. The route of bus 62 has exactly n stops (stop 1 goes first on its way and stop n goes last). The stops are positioned on a straight line and their coordinates are $0 = x_1 < x_2 < \dots < x_n$.

Each day exactly m people use bus 62. For each person we know the number of the stop where he gets on the bus and the number of the stop where he gets off the bus. A ticket from stop a to stop b ($a < b$) costs $x_b - x_a$ rubles. However, the conductor can choose no more than one segment NOT TO SELL a ticket for. We mean that conductor should choose C and D (C <= D) and sell a ticket for the segments $[A, C]$ and $[D, B]$, or not sell the ticket at all. The conductor and the passenger divide the saved money between themselves equally. The conductor's "untaxed income" is sometimes interrupted by inspections that take place as the bus drives on some segment of the route located between two consecutive stops. The inspector fines the conductor by c rubles for each passenger who doesn't have the ticket for this route's segment.

You know the coordinated of all stops x_i; the numbers of stops where the i-th passenger gets on and off, a_i and b_i ($a_i < b_i$); the fine c; and also p_i — the probability of inspection on segment between the i-th and the $i+1$-th stop. The conductor asked you to help him make a plan of selling tickets that maximizes the mathematical expectation of his profit.

Input

The first line contains three integers n, m and c ($2 \le n \le 1.5 \cdot 10^5$, $1 \le m \le 3 \cdot 10^5$, $1 \le c \le 10^4$).

The next line contains n integers x_i ($0 \le x_i \le 10^9$, $x_1 = 0$, $x_i < x_{i+1}$) — the coordinates of the stops on the bus's route.

The third line contains $n - 1$ integer p_i ($0 \le p_i \le 100$) — the probability of inspection in percents on the segment between stop i and stop $i + 1$.

Then follow m lines that describe the bus's passengers. Each line contains exactly two integers a_i and b_i ($1 \le a_i < b_i \le n$) — the numbers of stops where the i-th passenger gets on and off.

Output

Print the single real number — the maximum expectation of the conductor's profit. The answer will be considered correct if its relative or absolute error does not exceed 10^{-6}.

Sample test(s)

input

```
3 3 10
0 10 100
100 0
1 2
2 3
1 3
```

output

```
90.000000000
```

input

```
10 8 187
0 10 30 70 150 310 630 1270 2550 51100
13 87 65 0 100 44 67 3 4
1 10
2 9
3 8
1 5
6 10
2 7
4 10
4 5
```

output

```
76859.990000000
```

Note

A comment to the first sample:

The first and third passengers get tickets from stop 1 to stop 2. The second passenger doesn't get a ticket. There always is inspection on the segment 1-2 but both passengers have the ticket for it. There never is an inspection on the segment 2-3, that's why the second passenger gets away with the cheating. Our total profit is $(0 + 90 / 2 + 90 / 2) = 90$.