A1. Beaver's Calculator 1.0

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

The Smart Beaver from ABBYY has once again surprised us! He has developed a new calculating device, which he called the "Beaver's Calculator 1.0". It is very peculiar and it is planned to be used in a variety of scientific problems.

To test it, the Smart Beaver invited n scientists, numbered from 1 to n. The i-th scientist brought k_i calculating problems for the device developed by the Smart Beaver from ABBYY. The problems of the i-th scientist are numbered from 1 to k_i, and they must be calculated sequentially in the described order, since calculating each problem heavily depends on the results of calculating of the previous ones.

Each problem of each of the n scientists is described by one integer $a_{i, j}$, where i $(1 \le i \le n)$ is the number of the scientist, j $(1 \le j \le k_i)$ is the number of the problem, and $a_{i, j}$ is the number of resource units the calculating device needs to solve this problem.

The calculating device that is developed by the Smart Beaver is pretty unusual. It solves problems sequentially, one after another. After some problem is solved and before the next one is considered, the calculating device allocates or frees resources.

The most expensive operation for the calculating device is freeing resources, which works much slower than allocating them. It is therefore desirable that each next problem for the calculating device requires no less resources than the previous one.

You are given the information about the problems the scientists offered for the testing. You need to arrange these problems in such an order that the number of adjacent "bad" pairs of problems in this list is minimum possible. We will call two consecutive problems in this list a "bad pair" if the problem that is performed first requires more resources than the one that goes after it. Do not forget that the problems of the same scientist must be solved in a fixed order.

Input

The first line contains integer n — the number of scientists. To lessen the size of the input, each of the next n lines contains five integers k_i, $a_{i, 1}$, x_i, y_i, m_i $(0 \le a_{i, 1} < m_i \le 10^9$, $1 \le x_i, y_i \le 10^9)$ — the number of problems of the i-th scientist, the resources the first problem requires and three parameters that generate the subsequent values of $a_{i, j}$. For all j from 2 to k_i, inclusive, you should calculate value $a_{i, j}$ by formula $a_{i, j} = (a_{i, j-1} * x_i + y_i) \bmod m_i$, where $a \bmod b$ is the operation of taking the remainder of division of number a by number b.

To get the full points for the first group of tests it is sufficient to solve the problem with $n = 2$, $1 \le k_i \le 2000$.

To get the full points for the second group of tests it is sufficient to solve the problem with $n = 2$, $1 \le k_i \le 200000$.

To get the full points for the third group of tests it is sufficient to solve the problem with $1 \le n \le 5000$, $1 \le k_i \le 5000$.

Output

On the first line print a single number — the number of "bad" pairs in the optimal order.

If the total number of problems does not exceed 200000, also print $\sum k_i$ lines — the optimal order of the problems. On each of these lines print two integers separated by a single space — the required number of resources for the problem and the number of the scientist who offered this problem, respectively. The scientists are numbered from 1 to n in the order of input.

Examples

input

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

output

```
0
1 1
2 1
3 2
4 2
```

input

```
2
3 10 2 3 1000
```

```
3 100 1 999 1000
```

output

```
2
10 1
23 1
49 1
100 2
99 2
98 2
```

Note

In the first sample $n = 2$, $k_1 = 2$, $a_{1,1} = 1$, $a_{1,2} = 2$, $k_2 = 2$, $a_{2,1} = 3$, $a_{2,2} = 4$. We've got two scientists, each of them has two calculating problems. The problems of the first scientist require 1 and 2 resource units, the problems of the second one require 3 and 4 resource units. Let's list all possible variants of the calculating order (each problem is characterized only by the number of resource units it requires): $(1, 2, 3, 4)$, $(1, 3, 2, 4)$, $(3, 1, 2, 4)$, $(1, 3, 4, 2)$, $(3, 4, 1, 2)$, $(3, 1, 4, 2)$.

Sequence of problems $(1, 3, 2, 4)$ has one "bad" pair (3 and 2), $(3, 1, 4, 2)$ has two "bad" pairs (3 and 1, 4 and 2), and $(1, 2, 3, 4)$ has no "bad" pairs.

A2. Beaver's Calculator 1.0

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

The Smart Beaver from ABBYY has once again surprised us! He has developed a new calculating device, which he called the "Beaver's Calculator 1.0". It is very peculiar and it is planned to be used in a variety of scientific problems.

To test it, the Smart Beaver invited n scientists, numbered from 1 to n. The i-th scientist brought k_i calculating problems for the device developed by the Smart Beaver from ABBYY. The problems of the i-th scientist are numbered from 1 to k_i, and they must be calculated sequentially in the described order, since calculating each problem heavily depends on the results of calculating of the previous ones.

Each problem of each of the n scientists is described by one integer $a_{i, j}$, where i ($1 \le i \le n$) is the number of the scientist, j ($1 \le j \le k_i$) is the number of the problem, and $a_{i, j}$ is the number of resource units the calculating device needs to solve this problem.

The calculating device that is developed by the Smart Beaver is pretty unusual. It solves problems sequentially, one after another. After some problem is solved and before the next one is considered, the calculating device allocates or frees resources.

The most expensive operation for the calculating device is freeing resources, which works much slower than allocating them. It is therefore desirable that each next problem for the calculating device requires no less resources than the previous one.

You are given the information about the problems the scientists offered for the testing. You need to arrange these problems in such an order that the number of adjacent "bad" pairs of problems in this list is minimum possible. We will call two consecutive problems in this list a "bad pair" if the problem that is performed first requires more resources than the one that goes after it. Do not forget that the problems of the same scientist must be solved in a fixed order.

Input

The first line contains integer n — the number of scientists. To lessen the size of the input, each of the next n lines contains five integers k_i, $a_{i, 1}$, x_i, y_i, m_i ($0 \le a_{i, 1} < m_i \le 10^9$, $1 \le x_i, y_i \le 10^9$) — the number of problems of the i-th scientist, the resources the first problem requires and three parameters that generate the subsequent values of $a_{i, j}$. For all j from 2 to k_i, inclusive, you should calculate value $a_{i, j}$ by formula $a_{i, j} = (a_{i, j - 1} * x_i + y_i) \ mod \ m_i$, where $a \ mod \ b$ is the operation of taking the remainder of division of number a by number b.

To get the full points for the first group of tests it is sufficient to solve the problem with $n = 2$, $1 \le k_i \le 2000$.

To get the full points for the second group of tests it is sufficient to solve the problem with $n = 2$, $1 \le k_i \le 200000$.

To get the full points for the third group of tests it is sufficient to solve the problem with $1 \le n \le 5000$, $1 \le k_i \le 5000$.

Output

On the first line print a single number — the number of "bad" pairs in the optimal order.

If the total number of problems does not exceed 200000, also print $\sum k_i$ lines — the optimal order of the problems. On each of these lines print two integers separated by a single space — the required number of resources for the problem and the number of the scientist who offered this problem, respectively. The scientists are numbered from 1 to n in the order of input.

Examples

input

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

output

```
0
1 1
2 1
3 2
4 2
```

input

```
2
3 10 2 3 1000
3 100 1 999 1000
```

output

```
2
10 1
23 1
49 1
100 2
```

```
99 2
98 2
```

Note

In the first sample $n = 2$, $k_1 = 2$, $a_{1,1} = 1$, $a_{1,2} = 2$, $k_2 = 2$, $a_{2,1} = 3$, $a_{2,2} = 4$. We've got two scientists, each of them has two calculating problems. The problems of the first scientist require 1 and 2 resource units, the problems of the second one require 3 and 4 resource units. Let's list all possible variants of the calculating order (each problem is characterized only by the number of resource units it requires): $(1, 2, 3, 4)$, $(1, 3, 2, 4)$, $(3, 1, 2, 4)$, $(1, 3, 4, 2)$, $(3, 4, 1, 2)$, $(3, 1, 4, 2)$.

Sequence of problems $(1, 3, 2, 4)$ has one "bad" pair (3 and 2), $(3, 1, 4, 2)$ has two "bad" pairs (3 and 1, 4 and 2), and $(1, 2, 3, 4)$ has no "bad" pairs.

A3. Beaver's Calculator 1.0

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

The Smart Beaver from ABBYY has once again surprised us! He has developed a new calculating device, which he called the "Beaver's Calculator 1.0". It is very peculiar and it is planned to be used in a variety of scientific problems.

To test it, the Smart Beaver invited n scientists, numbered from 1 to n. The i-th scientist brought k_i calculating problems for the device developed by the Smart Beaver from ABBYY. The problems of the i-th scientist are numbered from 1 to k_i, and they must be calculated sequentially in the described order, since calculating each problem heavily depends on the results of calculating of the previous ones.

Each problem of each of the n scientists is described by one integer $a_{i, j}$, where i $(1 \le i \le n)$ is the number of the scientist, j $(1 \le j \le k_i)$ is the number of the problem, and $a_{i, j}$ is the number of resource units the calculating device needs to solve this problem.

The calculating device that is developed by the Smart Beaver is pretty unusual. It solves problems sequentially, one after another. After some problem is solved and before the next one is considered, the calculating device allocates or frees resources.

The most expensive operation for the calculating device is freeing resources, which works much slower than allocating them. It is therefore desirable that each next problem for the calculating device requires no less resources than the previous one.

You are given the information about the problems the scientists offered for the testing. You need to arrange these problems in such an order that the number of adjacent "bad" pairs of problems in this list is minimum possible. We will call two consecutive problems in this list a "bad pair" if the problem that is performed first requires more resources than the one that goes after it. Do not forget that the problems of the same scientist must be solved in a fixed order.

Input

The first line contains integer n — the number of scientists. To lessen the size of the input, each of the next n lines contains five integers k_i, $a_{i, 1}$, x_i, y_i, m_i $(0 \le a_{i, 1} < m_i \le 10^9, 1 \le x_i, y_i \le 10^9)$ — the number of problems of the i-th scientist, the resources the first problem requires and three parameters that generate the subsequent values of $a_{i, j}$. For all j from 2 to k_i, inclusive, you should calculate value $a_{i, j}$ by formula $a_{i, j} = (a_{i, j - 1} * x_i + y_i) \bmod m_i$, where $a \bmod b$ is the operation of taking the remainder of division of number a by number b.

To get the full points for the first group of tests it is sufficient to solve the problem with $n = 2, 1 \le k_i \le 2000$.

To get the full points for the second group of tests it is sufficient to solve the problem with $n = 2, 1 \le k_i \le 200000$.

To get the full points for the third group of tests it is sufficient to solve the problem with $1 \le n \le 5000, 1 \le k_i \le 5000$.

Output

On the first line print a single number — the number of "bad" pairs in the optimal order.

If the total number of problems does not exceed 200000, also print $\sum k_i$ lines — the optimal order of the problems. On each of these lines print two integers separated by a single space — the required number of resources for the problem and the number of the scientist who offered this problem, respectively. The scientists are numbered from 1 to n in the order of input.

Examples

input

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

output

```
0
1 1
2 1
3 2
4 2
```

input

```
2
3 10 2 3 1000
3 100 1 999 1000
```

output

```
2
10 1
23 1
49 1
100 2
```

```
99 2
98 2
```

Note

In the first sample $n = 2$, $k_1 = 2$, $a_{1,1} = 1$, $a_{1,2} = 2$, $k_2 = 2$, $a_{2,1} = 3$, $a_{2,2} = 4$. We've got two scientists, each of them has two calculating problems. The problems of the first scientist require 1 and 2 resource units, the problems of the second one require 3 and 4 resource units. Let's list all possible variants of the calculating order (each problem is characterized only by the number of resource units it requires): $(1, 2, 3, 4)$, $(1, 3, 2, 4)$, $(3, 1, 2, 4)$, $(1, 3, 4, 2)$, $(3, 4, 1, 2)$, $(3, 1, 4, 2)$.

Sequence of problems $(1, 3, 2, 4)$ has one "bad" pair (3 and 2), $(3, 1, 4, 2)$ has two "bad" pairs (3 and 1, 4 and 2), and $(1, 2, 3, 4)$ has no "bad" pairs.

B1. Military Trainings

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

The Smart Beaver from ABBYY started cooperating with the Ministry of Defence. Now they train soldiers to move armoured columns. The training involves testing a new type of tanks that can transmit information. To test the new type of tanks, the training has a special exercise, its essence is as follows.

Initially, the column consists of n tanks sequentially numbered from 1 to n in the order of position in the column from its beginning to its end. During the whole exercise, exactly n messages must be transferred from the beginning of the column to its end.

Transferring one message is as follows. The tank that goes first in the column transmits the message to some tank in the column. The tank which received the message sends it further down the column. The process is continued until the last tank receives the message. It is possible that not all tanks in the column will receive the message — it is important that the last tank in the column should receive the message.

After the last tank (tank number n) receives the message, it moves to the beginning of the column and sends another message to the end of the column in the same manner. When the message reaches the last tank (tank number $n - 1$), that tank moves to the beginning of the column and sends the next message to the end of the column, and so on. Thus, the exercise is completed when the tanks in the column return to their original order, that is, immediately after tank number 1 moves to the beginning of the column.

If the tanks were initially placed in the column in the order $1, 2, ..., n$, then after the first message their order changes to $n, 1, ..., n - 1$, after the second message it changes to $n - 1, n, 1, ..., n - 2$, and so on.

The tanks are constructed in a very peculiar way. The tank with number i is characterized by one integer a_i, which is called the *message receiving radius* of this tank.

Transferring a message between two tanks takes one second, however, not always one tank can transmit a message to another one. Let's consider two tanks in the column such that the first of them is the i-th in the column counting from the beginning, and the second one is the j-th in the column, and suppose the second tank has number x. Then the first tank can transmit a message to the second tank if $i < j$ and $i \geq j - a_x$.

The Ministry of Defense (and soon the Smart Beaver) faced the question of how to organize the training efficiently. The exercise should be finished as quickly as possible. We'll neglect the time that the tanks spend on moving along the column, since improving the tanks' speed is not a priority for this training.

You are given the number of tanks, as well as the message receiving radii of all tanks. You must help the Smart Beaver and organize the transferring of messages in a way that makes the total transmission time of all messages as small as possible.

Input

The first line contains integer n — the number of tanks in the column. Each of the next n lines contains one integer a_i ($1 \leq a_i \leq 250000$, $1 \leq i \leq n$) — the message receiving radii of the tanks in the order from tank 1 to tank n (let us remind you that initially the tanks are located in the column in ascending order of their numbers).

To get the full points for the first group of tests it is sufficient to solve the problem with $2 \leq n \leq 300$.

To get the full points for the second group of tests it is sufficient to solve the problem with $2 \leq n \leq 10000$.

To get the full points for the third group of tests it is sufficient to solve the problem with $2 \leq n \leq 250000$.

Output

Print a single integer — the minimum possible total time of transmitting the messages.

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

Examples

input

```
3
2
1
1
```

output

```
5
```

input

```
5
2
2
```

```
2
2
2
```

output

```
10
```

Note

In the first sample the original order of tanks is $1, 2, 3$. The first tank sends a message to the second one, then the second tank sends it to the third one — it takes two seconds. The third tank moves to the beginning of the column and the order of tanks now is $3, 1, 2$. The third tank sends a message to the first one, then the first one sends it to the second one — it takes two more seconds. The second tank moves to the beginning and the order of the tanks is now $2, 3, 1$. With this arrangement, the second tank can immediately send a message to the first one, since the message receiving radius of the first tank is large enough — it takes one second. Finally, the tanks return to their original order $1, 2, 3$. In total, the exercise takes 5 seconds.

In the second sample, all five tanks are the same and sending a single message takes two seconds, so in total the exercise takes 10 seconds.

B2. Military Trainings

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

The Smart Beaver from ABBYY started cooperating with the Ministry of Defence. Now they train soldiers to move armoured columns. The training involves testing a new type of tanks that can transmit information. To test the new type of tanks, the training has a special exercise, its essence is as follows.

Initially, the column consists of n tanks sequentially numbered from 1 to n in the order of position in the column from its beginning to its end. During the whole exercise, exactly n messages must be transferred from the beginning of the column to its end.

Transferring one message is as follows. The tank that goes first in the column transmits the message to some tank in the column. The tank which received the message sends it further down the column. The process is continued until the last tank receives the message. It is possible that not all tanks in the column will receive the message — it is important that the last tank in the column should receive the message.

After the last tank (tank number n) receives the message, it moves to the beginning of the column and sends another message to the end of the column in the same manner. When the message reaches the last tank (tank number $n - 1$), that tank moves to the beginning of the column and sends the next message to the end of the column, and so on. Thus, the exercise is completed when the tanks in the column return to their original order, that is, immediately after tank number 1 moves to the beginning of the column.

If the tanks were initially placed in the column in the order $1, 2, \ldots, n$, then after the first message their order changes to $n, 1, \ldots, n - 1$, after the second message it changes to $n - 1, n, 1, \ldots, n - 2$, and so on.

The tanks are constructed in a very peculiar way. The tank with number i is characterized by one integer a_i, which is called the *message receiving radius* of this tank.

Transferring a message between two tanks takes one second, however, not always one tank can transmit a message to another one. Let's consider two tanks in the column such that the first of them is the i-th in the column counting from the beginning, and the second one is the j-th in the column, and suppose the second tank has number x. Then the first tank can transmit a message to the second tank if $i < j$ and $i \geq j - a_x$.

The Ministry of Defense (and soon the Smart Beaver) faced the question of how to organize the training efficiently. The exercise should be finished as quickly as possible. We'll neglect the time that the tanks spend on moving along the column, since improving the tanks' speed is not a priority for this training.

You are given the number of tanks, as well as the message receiving radii of all tanks. You must help the Smart Beaver and organize the transferring of messages in a way that makes the total transmission time of all messages as small as possible.

Input

The first line contains integer n — the number of tanks in the column. Each of the next n lines contains one integer a_i ($1 \leq a_i \leq 250000$, $1 \leq i \leq n$) — the message receiving radii of the tanks in the order from tank 1 to tank n (let us remind you that initially the tanks are located in the column in ascending order of their numbers).

To get the full points for the first group of tests it is sufficient to solve the problem with $2 \leq n \leq 300$.

To get the full points for the second group of tests it is sufficient to solve the problem with $2 \leq n \leq 10000$.

To get the full points for the third group of tests it is sufficient to solve the problem with $2 \leq n \leq 250000$.

Output

Print a single integer — the minimum possible total time of transmitting the messages.

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

Examples

input

```
3
2
1
1
```

output

```
5
```

input

```
5
2
2
```

```
2
2
2
```

output

```
10
```

Note

In the first sample the original order of tanks is $1, 2, 3$. The first tank sends a message to the second one, then the second tank sends it to the third one — it takes two seconds. The third tank moves to the beginning of the column and the order of tanks now is $3, 1, 2$. The third tank sends a message to the first one, then the first one sends it to the second one — it takes two more seconds. The second tank moves to the beginning and the order of the tanks is now $2, 3, 1$. With this arrangement, the second tank can immediately send a message to the first one, since the message receiving radius of the first tank is large enough — it takes one second. Finally, the tanks return to their original order $1, 2, 3$. In total, the exercise takes 5 seconds.

In the second sample, all five tanks are the same and sending a single message takes two seconds, so in total the exercise takes 10 seconds.

B3. Military Trainings

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

The Smart Beaver from ABBYY started cooperating with the Ministry of Defence. Now they train soldiers to move armoured columns. The training involves testing a new type of tanks that can transmit information. To test the new type of tanks, the training has a special exercise, its essence is as follows.

Initially, the column consists of n tanks sequentially numbered from 1 to n in the order of position in the column from its beginning to its end. During the whole exercise, exactly n messages must be transferred from the beginning of the column to its end.

Transferring one message is as follows. The tank that goes first in the column transmits the message to some tank in the column. The tank which received the message sends it further down the column. The process is continued until the last tank receives the message. It is possible that not all tanks in the column will receive the message — it is important that the last tank in the column should receive the message.

After the last tank (tank number n) receives the message, it moves to the beginning of the column and sends another message to the end of the column in the same manner. When the message reaches the last tank (tank number $n - 1$), that tank moves to the beginning of the column and sends the next message to the end of the column, and so on. Thus, the exercise is completed when the tanks in the column return to their original order, that is, immediately after tank number 1 moves to the beginning of the column.

If the tanks were initially placed in the column in the order $1, 2, ..., n$, then after the first message their order changes to $n, 1, ..., n - 1$, after the second message it changes to $n - 1, n, 1, ..., n - 2$, and so on.

The tanks are constructed in a very peculiar way. The tank with number i is characterized by one integer a_i, which is called the *message receiving radius* of this tank.

Transferring a message between two tanks takes one second, however, not always one tank can transmit a message to another one. Let's consider two tanks in the column such that the first of them is the i-th in the column counting from the beginning, and the second one is the j-th in the column, and suppose the second tank has number x. Then the first tank can transmit a message to the second tank if $i < j$ and $i \geq j - a_x$.

The Ministry of Defense (and soon the Smart Beaver) faced the question of how to organize the training efficiently. The exercise should be finished as quickly as possible. We'll neglect the time that the tanks spend on moving along the column, since improving the tanks' speed is not a priority for this training.

You are given the number of tanks, as well as the message receiving radii of all tanks. You must help the Smart Beaver and organize the transferring of messages in a way that makes the total transmission time of all messages as small as possible.

Input

The first line contains integer n — the number of tanks in the column. Each of the next n lines contains one integer a_i ($1 \leq a_i \leq 250000$, $1 \leq i \leq n$) — the message receiving radii of the tanks in the order from tank 1 to tank n (let us remind you that initially the tanks are located in the column in ascending order of their numbers).

To get the full points for the first group of tests it is sufficient to solve the problem with $2 \leq n \leq 300$.

To get the full points for the second group of tests it is sufficient to solve the problem with $2 \leq n \leq 10000$.

To get the full points for the third group of tests it is sufficient to solve the problem with $2 \leq n \leq 250000$.

Output

Print a single integer — the minimum possible total time of transmitting the messages.

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

Examples

input

```
3
2
1
1
```

output

```
5
```

input

```
5
2
2
```

```
2
2
2
```

output

```
10
```

Note

In the first sample the original order of tanks is $1, 2, 3$. The first tank sends a message to the second one, then the second tank sends it to the third one — it takes two seconds. The third tank moves to the beginning of the column and the order of tanks now is $3, 1, 2$. The third tank sends a message to the first one, then the first one sends it to the second one — it takes two more seconds. The second tank moves to the beginning and the order of the tanks is now $2, 3, 1$. With this arrangement, the second tank can immediately send a message to the first one, since the message receiving radius of the first tank is large enough — it takes one second. Finally, the tanks return to their original order $1, 2, 3$. In total, the exercise takes 5 seconds.

In the second sample, all five tanks are the same and sending a single message takes two seconds, so in total the exercise takes 10 seconds.

C1. Game with Two Trees

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

The Smart Beaver from ABBYY has come up with a new developing game for children. The Beaver thinks that this game will help children to understand programming better.

The main object of the game is finite rooted trees, each of their edges contains some lowercase English letter. Vertices on any tree are always numbered sequentially from 1 to m, where m is the number of vertices in the tree. Before describing the actual game, let's introduce some definitions.

We'll assume that the sequence of vertices with numbers $v_1, v_2, \dots, v_k$ ($k \ge 1$) is a *forward path*, if for any integer i from 1 to $k - 1$ vertex v_i is a direct ancestor of vertex v_{i+1}. If we sequentially write out all letters from the the edges of the given path from v_1 to v_k, we get some string ($k = 1$ gives us an empty string). We'll say that such string corresponds to forward path $v_1, v_2, \dots, v_k$.

We'll assume that the sequence of tree vertices with numbers $v_1, v_2, \dots, v_k$ ($k \ge 1$) is a *backward path* if for any integer i from 1 to $k - 1$ vertex v_i is the direct descendant of vertex v_{i+1}. If we sequentially write out all the letters from the edges of the given path from v_1 to v_k, we get some string ($k = 1$ gives us an empty string). We'll say that such string corresponds to backward path $v_1, v_2, \dots, v_k$.

Now let's describe the game that the Smart Beaver from ABBYY has come up with. The game uses two rooted trees, each of which initially consists of one vertex with number 1. The player is given some sequence of operations. Each operation is characterized by three values (t, v, c) where:

- t is the number of the tree on which the operation is executed (1 or 2);
- v is the vertex index in this tree (it is guaranteed that the tree contains a vertex with this index);
- c is a lowercase English letter.

The actual operation is as follows: vertex v of tree t gets a new descendant with number $m + 1$ (where m is the current number of vertices in tree t), and there should be letter c put on the new edge from vertex v to vertex $m + 1$.

We'll say that an ordered group of three integers (i, j, q) is a *good combination* if:

- $1 \le i \le m_1$, where m_1 is the number of vertices in the first tree;
- $1 \le j, q \le m_2$, where m_2 is the number of vertices in the second tree;
- there exists a forward path $v_1, v_2, \dots, v_k$ such that $v_1 = j$ and $v_k = q$ in the second tree;
- the string that corresponds to the forward path in the second tree from vertex j to vertex q equals the string that corresponds to the backward path in the first tree from vertex i to vertex 1 (note that both paths are determined uniquely).

Your task is to calculate the number of existing good combinations after each operation on the trees.

Input

The first line contains integer n — the number of operations on the trees. Next n lines specify the operations in the order of their execution. Each line has form "t v c", where t is the number of the tree, v is the vertex index in this tree, and c is a lowercase English letter.

To get the full points for the first group of tests it is sufficient to solve the problem with $1 \le n \le 700$.

To get the full points for the second group of tests it is sufficient to solve the problem with $1 \le n \le 7000$.

To get the full points for the third group of tests it is sufficient to solve the problem with $1 \le n \le 100000$.

Output

Print exactly n lines, each containing one integer — the number of existing good combinations after the corresponding operation from the input.

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

Examples

input

```
5
1 1 a
2 1 a
1 2 b
2 1 b
2 3 a
```

output

```
1
3
3
4
7
```

Note

After the first operation the only good combination was $(1, 1, 1)$. After the second operation new good combinations appeared, $(2, 1, 2)$ and $(1, 2, 2)$. The third operation didn't bring any good combinations. The fourth operation added good combination $(1, 3, 3)$. Finally, the fifth operation resulted in as much as three new good combinations — $(1, 4, 4)$, $(2, 3, 4)$ and $(3, 1, 4)$.

C2. Game with Two Trees

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

The Smart Beaver from ABBYY has come up with a new developing game for children. The Beaver thinks that this game will help children to understand programming better.

The main object of the game is finite rooted trees, each of their edges contains some lowercase English letter. Vertices on any tree are always numbered sequentially from 1 to m, where m is the number of vertices in the tree. Before describing the actual game, let's introduce some definitions.

We'll assume that the sequence of vertices with numbers $v_1, v_2, \ldots, v_k$ ($k \geq 1$) is a *forward path*, if for any integer i from 1 to $k - 1$ vertex v_i is a direct ancestor of vertex v_{i+1}. If we sequentially write out all letters from the the edges of the given path from v_1 to v_k, we get some string ($k = 1$ gives us an empty string). We'll say that such string corresponds to forward path $v_1, v_2, \ldots, v_k$.

We'll assume that the sequence of tree vertices with numbers $v_1, v_2, \ldots, v_k$ ($k \geq 1$) is a *backward path* if for any integer i from 1 to $k - 1$ vertex v_i is the direct descendant of vertex v_{i+1}. If we sequentially write out all the letters from the edges of the given path from v_1 to v_k, we get some string ($k = 1$ gives us an empty string). We'll say that such string corresponds to backward path $v_1, v_2, \ldots, v_k$.

Now let's describe the game that the Smart Beaver from ABBYY has come up with. The game uses two rooted trees, each of which initially consists of one vertex with number 1. The player is given some sequence of operations. Each operation is characterized by three values (t, v, c) where:

- t is the number of the tree on which the operation is executed (1 or 2);
- v is the vertex index in this tree (it is guaranteed that the tree contains a vertex with this index);
- c is a lowercase English letter.

The actual operation is as follows: vertex v of tree t gets a new descendant with number $m + 1$ (where m is the current number of vertices in tree t), and there should be letter c put on the new edge from vertex v to vertex $m + 1$.

We'll say that an ordered group of three integers (i, j, q) is a *good combination* if:

- $1 \leq i \leq m_1$, where m_1 is the number of vertices in the first tree;
- $1 \leq j, q \leq m_2$, where m_2 is the number of vertices in the second tree;
- there exists a forward path $v_1, v_2, \ldots, v_k$ such that $v_1 = j$ and $v_k = q$ in the second tree;
- the string that corresponds to the forward path in the second tree from vertex j to vertex q equals the string that corresponds to the backward path in the first tree from vertex i to vertex 1 (note that both paths are determined uniquely).

Your task is to calculate the number of existing good combinations after each operation on the trees.

Input

The first line contains integer n — the number of operations on the trees. Next n lines specify the operations in the order of their execution. Each line has form "t v c", where t is the number of the tree, v is the vertex index in this tree, and c is a lowercase English letter.

To get the full points for the first group of tests it is sufficient to solve the problem with $1 \leq n \leq 700$.

To get the full points for the second group of tests it is sufficient to solve the problem with $1 \leq n \leq 7000$.

To get the full points for the third group of tests it is sufficient to solve the problem with $1 \leq n \leq 100000$.

Output

Print exactly n lines, each containing one integer — the number of existing good combinations after the corresponding operation from the input.

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

Examples

input

```
5
1 1 a
2 1 a
1 2 b
2 1 b
2 3 a
```

output

```
1
3
3
4
7
```

Note

After the first operation the only good combination was $(1, 1, 1)$. After the second operation new good combinations appeared, $(2, 1, 2)$ and $(1, 2, 2)$. The third operation didn't bring any good combinations. The fourth operation added good combination $(1, 3, 3)$. Finally, the fifth operation resulted in as much as three new good combinations — $(1, 4, 4)$, $(2, 3, 4)$ and $(3, 1, 4)$.

C3. Game with Two Trees

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

The Smart Beaver from ABBYY has come up with a new developing game for children. The Beaver thinks that this game will help children to understand programming better.

The main object of the game is finite rooted trees, each of their edges contains some lowercase English letter. Vertices on any tree are always numbered sequentially from 1 to m, where m is the number of vertices in the tree. Before describing the actual game, let's introduce some definitions.

We'll assume that the sequence of vertices with numbers $v_1, v_2, \ldots, v_k$ ($k \geq 1$) is a *forward path*, if for any integer i from 1 to $k - 1$ vertex v_i is a direct ancestor of vertex v_{i+1}. If we sequentially write out all letters from the the edges of the given path from v_1 to v_k, we get some string ($k = 1$ gives us an empty string). We'll say that such string corresponds to forward path $v_1, v_2, \ldots, v_k$.

We'll assume that the sequence of tree vertices with numbers $v_1, v_2, \ldots, v_k$ ($k \geq 1$) is a *backward path* if for any integer i from 1 to $k - 1$ vertex v_i is the direct descendant of vertex v_{i+1}. If we sequentially write out all the letters from the edges of the given path from v_1 to v_k, we get some string ($k = 1$ gives us an empty string). We'll say that such string corresponds to backward path $v_1, v_2, \ldots, v_k$.

Now let's describe the game that the Smart Beaver from ABBYY has come up with. The game uses two rooted trees, each of which initially consists of one vertex with number 1. The player is given some sequence of operations. Each operation is characterized by three values (t, v, c) where:

- t is the number of the tree on which the operation is executed (1 or 2);
- v is the vertex index in this tree (it is guaranteed that the tree contains a vertex with this index);
- c is a lowercase English letter.

The actual operation is as follows: vertex v of tree t gets a new descendant with number $m + 1$ (where m is the current number of vertices in tree t), and there should be letter c put on the new edge from vertex v to vertex $m + 1$.

We'll say that an ordered group of three integers (i, j, q) is a *good combination* if:

- $1 \leq i \leq m_1$, where m_1 is the number of vertices in the first tree;
- $1 \leq j, q \leq m_2$, where m_2 is the number of vertices in the second tree;
- there exists a forward path $v_1, v_2, \ldots, v_k$ such that $v_1 = j$ and $v_k = q$ in the second tree;
- the string that corresponds to the forward path in the second tree from vertex j to vertex q equals the string that corresponds to the backward path in the first tree from vertex i to vertex 1 (note that both paths are determined uniquely).

Your task is to calculate the number of existing good combinations after each operation on the trees.

Input

The first line contains integer n — the number of operations on the trees. Next n lines specify the operations in the order of their execution. Each line has form "t v c", where t is the number of the tree, v is the vertex index in this tree, and c is a lowercase English letter.

To get the full points for the first group of tests it is sufficient to solve the problem with $1 \leq n \leq 700$.

To get the full points for the second group of tests it is sufficient to solve the problem with $1 \leq n \leq 7000$.

To get the full points for the third group of tests it is sufficient to solve the problem with $1 \leq n \leq 100000$.

Output

Print exactly n lines, each containing one integer — the number of existing good combinations after the corresponding operation from the input.

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

Examples

input

```
5
1 1 a
2 1 a
1 2 b
2 1 b
2 3 a
```

output

```
1
3
3
4
7
```

Note

After the first operation the only good combination was $(1, 1, 1)$. After the second operation new good combinations appeared, $(2, 1, 2)$ and $(1, 2, 2)$. The third operation didn't bring any good combinations. The fourth operation added good combination $(1, 3, 3)$. Finally, the fifth operation resulted in as much as three new good combinations — $(1, 4, 4)$, $(2, 3, 4)$ and $(3, 1, 4)$.

D1. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \le id \le 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples

D10. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \le id \le 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples

D2. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \le id \le 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples

D3. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \le id \le 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples

D4. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \le id \le 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples

D5. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \le id \le 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples

D6. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \le id \le 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples

D7. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \leq id \leq 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples

D8. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \le id \le 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples

D9. The Beaver's Problem - 3

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

The Smart Beaver from ABBYY came up with another splendid problem for the ABBYY Cup participants! This time the Beaver invites the contest participants to check out a problem on sorting documents by their subjects. Let's describe the problem:

You've got some training set of documents. For each document you know its subject. The subject in this problem is an integer from 1 to 3. Each of these numbers has a physical meaning. For instance, all documents with subject 3 are about trade.

You can download the training set of documents at the following link: `http://download4.abbyy.com/a2/X2RZ2ZWXBG5VYWAL61H76ZQM/train.zip`. The archive contains three directories with names "1", "2", "3". Directory named "1" contains documents on the 1-st subject, directory "2" contains documents on the 2-nd subject, and directory "3" contains documents on the 3-rd subject. Each document corresponds to exactly one file from some directory.

All documents have the following format: the first line contains the document identifier, the second line contains the name of the document, all subsequent lines contain the text of the document. The document identifier is used to make installing the problem more convenient and has no useful information for the participants.

You need to write a program that should indicate the subject for a given document. It is guaranteed that all documents given as input to your program correspond to one of the three subjects of the training set.

Input

The first line contains integer id ($0 \le id \le 10^6$) — the document identifier. The second line contains the name of the document. The third and the subsequent lines contain the text of the document. It is guaranteed that the size of any given document will not exceed 10 kilobytes.

The tests for this problem are divided into 10 groups. Documents of groups 1 and 2 are taken from the training set, but their identifiers will not match the identifiers specified in the training set. Groups from the 3-rd to the 10-th are roughly sorted by the author in ascending order of difficulty (these groups contain documents which aren't present in the training set).

Output

Print an integer from 1 to 3, inclusive — the number of the subject the given document corresponds to.

Examples