Canada Cup 2016

A. Jumping Ball

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

In a new version of the famous Pinball game, one of the most important parts of the game field is a sequence of n bumpers. The bumpers are numbered with integers from 1 to n from left to right. There are two types of bumpers. They are denoted by the characters '<' and '>'. When the ball hits the bumper at position i it goes one position to the right (to the position $i + 1$) if the type of this bumper is '>', or one position to the left (to $i - 1$) if the type of the bumper at position i is '<'. If there is no such position, in other words if $i - 1 < 1$ or $i + 1 > n$, the ball falls from the game field.

Depending on the ball's starting position, the ball may eventually fall from the game field or it may stay there forever. You are given a string representing the bumpers' types. Calculate the number of positions such that the ball will eventually fall from the game field if it starts at that position.

Input

The first line of the input contains a single integer n ($1 \le n \le 200\,000$) — the length of the sequence of bumpers. The second line contains the string, which consists of the characters '<' and '>'. The character at the i-th position of this string corresponds to the type of the i-th bumper.

Output

Print one integer — the number of positions in the sequence such that the ball will eventually fall from the game field if it starts at that position.

Examples

input

```
4
<<><
```

output

```
2
```

input

```
5
>>>>>
```

output

```
5
```

input

```
4
>><<
```

output

```
0
```

Note

In the first sample, the ball will fall from the field if starts at position 1 or position 2.

In the second sample, any starting position will result in the ball falling from the field.

B. Food on the Plane

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

A new airplane SuperPuperJet has an infinite number of rows, numbered with positive integers starting with 1 from cockpit to tail. There are six seats in each row, denoted with letters from '`a`' to '`f`'. Seats '`a`', '`b`' and '`c`' are located to the left of an aisle (if one looks in the direction of the cockpit), while seats '`d`', '`e`' and '`f`' are located to the right. Seats '`a`' and '`f`' are located near the windows, while seats '`c`' and '`d`' are located near the aisle.

It's lunch time and two flight attendants have just started to serve food. They move from the first rows to the tail, always maintaining a distance of two rows from each other because of the food trolley. Thus, at the beginning the first attendant serves row 1 while the second attendant serves row 3. When both rows are done they move one row forward: the first attendant serves row 2 while the second attendant serves row 4. Then they move three rows forward and the first attendant serves row 5 while the second attendant serves row 7. Then they move one row forward again and so on.

Flight attendants work with the same speed: it takes exactly 1 second to serve one passenger and 1 second to move one row forward. Each attendant first serves the passengers on the seats to the right of the aisle and then serves passengers on the seats to the left of the aisle (if one looks in the direction of the cockpit). Moreover, they always serve passengers in order from the window to the aisle. Thus, the first passenger to receive food in each row is located in seat '`f`', and the last one — in seat '`c`'. Assume that all seats are occupied.

Vasya has seat s in row n and wants to know how many seconds will pass before he gets his lunch.

Input

The only line of input contains a description of Vasya's seat in the format ns, where n ($1 \leq n \leq 10^{18}$) is the index of the row and s is the seat in this row, denoted as letter from '`a`' to '`f`'. The index of the row and the seat **are not separated** by a space.

Output

Print one integer — the number of seconds Vasya has to wait until he gets his lunch.

Examples

input
```
1f
```
output
```
1
```

input
```
2d
```
output
```
10
```

input
```
4a
```
output
```
11
```

input
```
5e
```
output
```
18
```

Note

In the first sample, the first flight attendant serves Vasya first, so Vasya gets his lunch after 1 second.

In the second sample, the flight attendants will spend 6 seconds to serve everyone in the rows 1 and 3, then they will move one row forward in 1 second. As they first serve seats located to the right of the aisle in order from window to aisle, Vasya has to wait 3 more seconds. The total is $6 + 1 + 3 = 10$.

C. Hidden Word

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

Let's define a grid to be a set of tiles with 2 rows and 13 columns. Each tile has an English letter written in it. The letters don't have to be unique: there might be two or more tiles with the same letter written on them. Here is an example of a grid:

```
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
```

We say that two tiles are adjacent if they share a side or a corner. In the example grid above, the tile with the letter '`A`' is adjacent only to the tiles with letters '`B`', '`N`', and '`O`'. A tile is not adjacent to itself.

A sequence of tiles is called a path if each tile in the sequence is adjacent to the tile which follows it (except for the last tile in the sequence, which of course has no successor). In this example, "`ABC`" is a path, and so is "`KXWIHIJK`". "`MAB`" is not a path because '`M`' is not adjacent to '`A`'. A single tile can be used more than once by a path (though the tile cannot occupy two consecutive places in the path because no tile is adjacent to itself).

You're given a string s which consists of 27 upper-case English letters. Each English letter **occurs at least once** in s. Find a grid that contains a path whose tiles, viewed in the order that the path visits them, form the string s. If there's no solution, print "`Impossible`" (without the quotes).

Input

The only line of the input contains the string s, consisting of 27 upper-case English letters. Each English letter occurs at least once in s.

Output

Output two lines, each consisting of 13 upper-case English characters, representing the rows of the grid. If there are multiple solutions, print any of them. If there is no solution print "`Impossible`".

Examples

input

```
ABCDEFGHIJKLMNOPQRSGTUVWXYZ
```

output

```
YXWVUTGHIJKLM
ZABCDEFSRQPON
```

input

```
BUVTYZFQSNRIWOXXGJLKACPEMDH
```

output

```
Impossible
```

D. Contest Balloons

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

One tradition of ACM-ICPC contests is that a team gets a balloon for every solved problem. We assume that the submission time doesn't matter and teams are sorted only by the number of balloons they have. It means that one's place is equal to the number of teams with more balloons, increased by 1. For example, if there are seven teams with more balloons, you get the eight place. Ties are allowed.

You should know that it's important to eat before a contest. If the number of balloons of a team is greater than the weight of this team, the team starts to float in the air together with their workstation. They eventually touch the ceiling, what is strictly forbidden by the rules. The team is then disqualified and isn't considered in the standings.

A contest has just finished. There are n teams, numbered 1 through n. The i-th team has t_i balloons and weight w_i. It's guaranteed that t_i doesn't exceed w_i so nobody floats initially.

Limak is a member of the first team. He doesn't like cheating and he would never steal balloons from other teams. Instead, he can give his balloons away to other teams, possibly making them float. Limak can give away zero or more balloons of his team. Obviously, he can't give away more balloons than his team initially has.

What is the best place Limak can get?

Input

The first line of the standard input contains one integer n $(2 \le n \le 300\,000)$ — the number of teams.

The i-th of n following lines contains two integers t_i and w_i $(0 \le t_i \le w_i \le 10^{18})$ — respectively the number of balloons and the weight of the i-th team. Limak is a member of the first team.

Output

Print one integer denoting the best place Limak can get.

Examples

input
```
8
20 1000
32 37
40 1000
45 50
16 16
16 16
14 1000
2 1000
```
output
```
3
```

input
```
7
4 4
4 4
4 4
4 4
4 4
4 4
5 5
```
output
```
2
```

input
```
7
14000000003 1000000000000000000
81000000000 88000000000
5000000000 7000000000
15000000000 39000000000
46000000000 51000000000
0 1000000000
0 0
```
output
```
2
```

Note

In the first sample, Limak has 20 balloons initially. There are three teams with more balloons (32, 40 and 45 balloons), so Limak has the fourth place initially. One optimal strategy is:

1. Limak gives 6 balloons away to a team with 32 balloons and weight 37, which is just enough to make them fly. Unfortunately, Limak has only 14

balloons now and he would get the fifth place.
2. Limak gives 6 balloons away to a team with 45 balloons. Now they have 51 balloons and weight 50 so they fly and get disqualified.
3. Limak gives 1 balloon to each of two teams with 16 balloons initially.
4. Limak has $20 - 6 - 6 - 1 - 1 = 6$ balloons.
5. There are three other teams left and their numbers of balloons are 40, 14 and 2.
6. Limak gets the third place because there are two teams with more balloons.

In the second sample, Limak has the second place and he can't improve it.

In the third sample, Limak has just enough balloons to get rid of teams 2, 3 and 5 (the teams with $81\ 000\ 000\ 000$, $5\ 000\ 000\ 000$ and $46\ 000\ 000\ 000$ balloons respectively). With zero balloons left, he will get the second place (ex-aequo with team 6 and team 7).

E. Too Much Money

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

Alfred wants to buy a toy moose that costs c dollars. The store doesn’t give change, so he must give the store exactly c dollars, no more and no less. He has n coins. To make c dollars from his coins, he follows the following algorithm: let S be the set of coins being used. S is initially empty. Alfred repeatedly adds to S the highest-valued coin he has such that the total value of the coins in S after adding the coin doesn’t exceed c. If there is no such coin, and the value of the coins in S is still less than c, he gives up and goes home. Note that Alfred never removes a coin from S after adding it.

As a programmer, you might be aware that Alfred’s algorithm can fail even when there is a set of coins with value exactly c. For example, if Alfred has one coin worth \$3, one coin worth \$4, and two coins worth \$5, and the moose costs \$12, then Alfred will add both of the \$5 coins to S and then give up, since adding any other coin would cause the value of the coins in S to exceed \$12. Of course, Alfred could instead combine one \$3 coin, one \$4 coin, and one \$5 coin to reach the total.

Bob tried to convince Alfred that his algorithm was flawed, but Alfred didn’t believe him. Now Bob wants to give Alfred some coins (in addition to those that Alfred already has) such that Alfred’s algorithm fails. Bob can give Alfred any number of coins of any denomination (subject to the constraint that each coin must be worth a positive integer number of dollars). There can be multiple coins of a single denomination. He would like to minimize the total value of the coins he gives Alfred. Please find this minimum value. If there is no solution, print "`Greed is good`". You can assume that the answer, if it exists, is positive. In other words, Alfred's algorithm will work if Bob doesn't give him any coins.

Input

The first line contains c $(1 \le c \le 200\,000)$ — the price Alfred wants to pay. The second line contains n $(1 \le n \le 200\,000)$ — the number of coins Alfred initially has. Then n lines follow, each containing a single integer x $(1 \le x \le c)$ representing the value of one of Alfred's coins.

Output

If there is a solution, print the minimum possible total value of the coins in a solution. Otherwise, print "`Greed is good`" (without quotes).

Examples

input

```
12
3
5
3
4
```

output

```
5
```

input

```
50
8
1
2
4
8
16
37
37
37
```

output

```
Greed is good
```

Note

In the first sample, Bob should give Alfred a single coin worth \$5. This creates the situation described in the problem statement.

In the second sample, there is no set of coins that will cause Alfred's algorithm to fail.

F. Family Photos

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

Alice and Bonnie are sisters, but they don't like each other very much. So when some old family photos were found in the attic, they started to argue about who should receive which photos. In the end, they decided that they would take turns picking photos. Alice goes first.

There are n stacks of photos. Each stack contains **exactly two** photos. In each turn, a player may take only a photo from the top of one of the stacks.

Each photo is described by two non-negative integers a and b, indicating that it is worth a units of happiness to Alice and b units of happiness to Bonnie. Values of a and b might differ for different photos.

It's allowed to pass instead of taking a photo. The game ends when all photos are taken or both players pass consecutively.

The players don't act to maximize their own happiness. Instead, each player acts to maximize the amount by which her happiness exceeds her sister's. Assuming both players play optimal, find the difference between Alice's and Bonnie's happiness. That is, if there's a perfectly-played game such that Alice has x happiness and Bonnie has y happiness at the end, you should print x - y.

Input

The first line of input contains a single integer n ($1 \le n \le 100\,000$) — the number of two-photo stacks. Then follow n lines, each describing one of the stacks. A stack is described by four space-separated non-negative integers a_1, b_1, a_2 and b_2, each not exceeding 10^9. a_1 and b_1 describe the top photo in the stack, while a_2 and b_2 describe the bottom photo in the stack.

Output

Output a single integer: the difference between Alice's and Bonnie's happiness if both play optimally.

Examples

```
input
```
```
2
12 3 4 7
1 15 9 1
```
```
output
```
```
1
```

```
input
```
```
2
5 4 8 8
4 12 14 0
```
```
output
```
```
4
```

```
input
```
```
1
0 10 0 10
```
```
output
```
```
-10
```

G. Messages on a Tree

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

Alice and Bob are well-known for sending messages to each other. This time you have a rooted tree with Bob standing in the root node and copies of Alice standing in each of the other vertices. The root node has number 0, the rest are numbered 1 through n.

At some moments of time some copies of Alice want to send a message to Bob and receive an answer. We will call this copy the *initiator*. The process of sending a message contains several steps:

- The initiator sends the message to the person standing in the parent node and begins waiting for the answer.
- When some copy of Alice receives a message from some of her children nodes, she sends the message to the person standing in the parent node and begins waiting for the answer.
- When Bob receives a message from some of his child nodes, he immediately sends the answer to the child node where the message came from.
- When some copy of Alice (except for initiator) receives an answer she is waiting for, she immediately sends it to the child vertex where the message came from.
- When the initiator receives the answer she is waiting for, she doesn't send it to anybody.
- There is a special case: a copy of Alice can't wait for two answers at the same time, so if some copy of Alice receives a message from her child node while she already waits for some answer, she rejects the message and sends a message saying this back to the child node where the message came from. Then the copy of Alice in the child vertex processes this answer as if it was from Bob.
- The process of sending a message to a parent node or to a child node is instant but a receiver (a parent or a child) gets a message after 1 second.

If some copy of Alice receives several messages from child nodes at the same moment while she isn't waiting for an answer, she processes the message from the **initiator** with the smallest number and rejects all the rest. If some copy of Alice receives messages from children nodes and also receives the answer she is waiting for at the same instant, then Alice first processes the answer, then immediately continue as normal with the incoming messages.

You are given the moments of time when some copy of Alice becomes the initiator and sends a message to Bob. For each message, find the moment of time when the answer (either from Bob or some copy of Alice) will be received by the initiator.

You can assume that if Alice wants to send a message (i.e. become the initiator) while waiting for some answer, she immediately rejects the message and receives an answer from herself in no time.

Input

The first line of input contains two integers n and m $(1 \le n, m \le 200\ 000)$ — the number of nodes with Alices and the number of messages.

Second line contains n integers $p_1, p_2, ..., p_n$ $(0 \le p_i < i)$. The integer p_i is the number of the parent node of node i.

The next m lines describe the messages. The i-th of them contains two integers x_i and t_i $(1 \le x_i \le n,\ 1 \le t_i \le 10^9)$ — the number of the vertex of the initiator of the i-th message and the time of the initiation (in seconds). The messages are given in order of increasing initiation time (i.e. $t_{i+1} \ge t_i$ holds for $1 \le i < m$). The pairs (x_i, t_i) are distinct.

Output

Print m integers — the i-th of them is the moment of time when the answer for the i-th message will be received by the initiator.

Examples

input

```
6 3
0 1 2 3 2 5
4 6
6 9
5 11
```

output

```
14 13 11
```

input

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

output

```
5 3
```

input

```
8 3
0 1 1 2 3 3 4 5
6 1
8 2
4 5
```

output

```
7 6 11
```

Note

In the first example the first message is initiated at the moment 6, reaches Bob at the moment 10, and the answer reaches the initiator at the moment 14. The second message reaches vertex 2 at the moment 11. At this moment the copy of Alice in this vertex is still waiting for the answer for the first message, so she rejects the second message. The answer reaches the initiator at the moment 13. The third message is not sent at all, because at the moment 11 Alice in vertex 5 is waiting for the answer for the second message.

In the second example the first message reaches Bob, the second is rejected by Alice in vertex 1. This is because the message with smaller initiator number has the priority.

In the third example the first and the third messages reach Bob, while the second message is rejected by Alice in vertex 3.