acm icpc International Collegiate Programming Contest
icpc.foundation
TWO SIGMA icpc north america sponsor
JET BRAINS icpc global programming tools sponsor
2017 ACM ICPC Mid-Atlantic USA Regional Contest

DO NOT OPEN TILL CONTEST BEGINS

2017 ACM ICPC Mid-Atlantic North America Programming Contest

Problem Set

Nov. 11, 2017

Welcome to the 2017 ICPC Mid-Atlantic Regional.

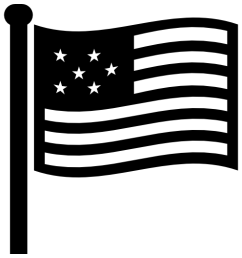
There are eight (8) problems in the packet, labeled A–H. These problems are NOT sorted by difficulty. As a team's solution is judged correct, the team will be awarded a balloon. The balloon colors are as follows:

Problem	Problem Name	Balloon Color
A	Star Arrangement	Orange
B	Security Badge	Green
C	Purple Rain	Silver
D	Avoiding Airports	Pink
E	Rainbow Roads	Red
F	Spinning Up Palindromes	Yellow
G	Long, Long Strings	Black
H	Haiku Formatting	Purple

Please be sure that you have read and understand the **Contest Guide and Rules**, provided separately.

Good luck, and HAVE FUN!!!

Problem A: Star Arrangements



The recent vote in Puerto Rico favoring United States statehood has made flag makers very excited. An updated flag with 51 stars rather than the current one with 50 would cause a huge jump in U.S. flag sales. The current pattern for 50 stars is five rows of 6 stars, interlaced with four offset rows of 5 stars. The rows alternate until all stars are represented.

```
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
```

This pattern has the property that adjacent rows differ by no more than one star. We represent this star arrangement compactly by the number of stars in the first two rows: `6,5`.

A 51-star flag that has the same property can have three rows of 9 stars, interlaced with three rows of 8 stars (with a compact representation of `9,8`). Conversely, if a state were to leave the union, one appealing representation would be seven rows of seven stars (`7,7`).

A flag pattern is *visually appealing* if it satisfies the following conditions:

- Every other row has the same number of stars.
- Adjacent rows differ by no more than one star.
- The first row cannot have fewer stars than the second row.

Your team sees beyond the short-term change to 51 for the US flag. You want to corner the market on flags for any union of three or more states. Given the number S of stars to draw on a flag, find all possible *visually appealing* flag patterns.

Input

The input consists of a single line containing the integer S ($3 \leq S \leq 32\,767$).

Output

On the first line, print S, followed by a colon. Then, for each visually appealing flag of S stars, print its compact representation, one per line.

This list of compact representations should be printed in increasing order of the number of stars in the first row; if there are ties, print them in order of the number of stars in the second row. The cases 1-by-S and S-by-1 are trivial, so do not print those arrangements.

The compact representations must be printed in the form "`x,y`", with exactly one comma between `x` and `y` and no other characters.

Examples

Example 1

Sample Input

```
3
```

Sample Output

```
3:
2,1
```

Example 2

Sample Input

```
50
```

Sample Output

```
50:
2,1
2,2
3,2
5,4
5,5
6,5
10,10
13,12
17,16
25,25
```

Example 3

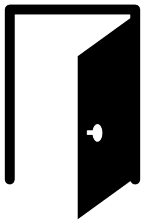
Sample Input

```
51
```

Sample Output

```
51:
2,1
3,3
9,8
17,17
26,25
```

Problem B: Security Badge



The home office of Labyrinthian Inc. has installed a new system of security badges and electronic door locks. Each badge is assigned an ID number, and the idea was that electronic locks on each door should allow access only to personnel whose ID number indicated that they had appropriate security clearance to enter that room, hallway, closet, or whatever lay on the other side of the door.

The contract for the lock system, however, was put out to the lowest bidder, who clearly misunderstood the intention. Instead of each lock storing a list of permitted ID numbers, instead each lock stores exactly two numbers, a lower and upper bound, and permits passage to badges whose number lies between those bounds. For example, a lock keyed as $(25, 29)$ would pass only badges $25, 26, 27, 28,$ and 29.

Complicating the matter is the fact that lock on each side of the door can be keyed differently, so a person who is able to pass through the door in one direction might not be able to return once the door has closed behind them.

The results have been frustrating (and occasionally entertaining – videos of everyone in the company trying to find a way to the cafeteria at noon have gone viral on social media).

It has become a major task, when hiring or promoting any employee, to find a badge number that will actually get them from the front door to their new office.

Write a program to determine how many badge numbers will permit passage from one given room to another.

Input

The first line of input will contain integers N, L, and B, denoting the number of rooms, of locks, and of badge numbers, respectively. $2 \leq N \leq 1\,000$, $1 \leq L \leq 5\,000$, $1 \leq B \leq 10^9$

The next line of input will contain two integers, S and D, $1 \leq S \leq N$, $1 \leq D \leq N$, $S \neq D$, denoting the starting and destination rooms that we are interested in.

This is followed by L lines, each describing a lock as four integers:

$$a\ b\ x\ y$$

indicating that a lock permits passage from room a to room b (but not from b to a) for badges numbered from x to y, inclusive. It is guaranteed that $1 \leq a, b \leq N$, $a \neq b$, $1 \leq x \leq B$, $1 \leq y \leq B$, $x \leq y$, and no (a, b) pair will occur more than once, although both (a, b) and (b, a) may occur within separate lines.

Output

Print a single line indicating the number of badge ID numbers permitting passage from room S to room D

Time limit: solutions should complete in no more than 5 CPU seconds

Examples

Example 1

Sample Input

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

Sample Output

```
5
```

Example 2

Sample Input

```
4 5 9
1 4
1 2 3 5
1 3 6 7
1 4 2 3
2 4 4 6
3 4 7 9
```

Sample Output

```
5
```

Problem C: Purple Rain

Purple rain falls in the magic kingdom of Linearland which is a straight, thin peninsula.

On close observation however, Prof. Nelson Rogers finds that actually it is a mix of Red and Blue drops.

In his zeal, he records the location and color of the raindrops in different locations along the peninsula. Looking at the data, Professor Rogers wants to know which part of Linearland had the "least" purple rain.

After some thought, he decides to model this problem as follows. Divide the peninsula into n sections and number them West to East from 1 to n. Then, describe the raindrops as a sequence of `R` and `B`, depending on whether the rainfall in each section is primarily red or blue. Finally, find a subsequence of where the difference between the number of `R` and the number of `B` is maximized.

Input

The input consists of a single line containing a string of n characters ($1 \leq n \leq 10^5$), describing the color of the raindrops in sections 1 to n.

It is guaranteed that the string consists of uppercase ASCII letters '`R`' and '`B`' only.

Output

Print, on a single line, two space-separated integers that describe the starting and ending positions of the part of Linearland that had the least purple rain.

If there are multiple possible answers, print the one that has the Westernmost (smallest-numbered) starting section. If there are multiple answers with the same Westernmost starting section, print the one with the Westernmost ending section.

Examples

Example 1

Sample Input

```
BBRRBRRBRB
```

Sample Output

```
3 7
```

Example 2

Sample Input

```
BBRBBRRB
```

Sample Output

```
1 5
```

Problem D: Avoiding Airports

David is looking to book some travel over the world. There are n countries that he can visit, and m flights that are available. The ith flight goes from country a_i to country b_i. It departs at time s_i, and lands at time e_i.

David is currently at the airport in country 1, and the current time is 0, and he would like to travel country n. He does not care about the total amount of time needed to travel, but he really hates waiting in the airport. If he waits t seconds in an airport, he gains t^2 units of frustration. Help him find an itinerary that minimizes the sum of frustration.

Input

The first line of input contains two space-separated integers n and m ($1 \leq n, m \leq 200\,000$).

Each of the next m lines contains four space-separated integers a_i, b_i, s_i, and e_i ($1 \leq a_i, b_i \leq n$; $0 \leq s_i \leq e_i \leq 10^6$).

A flight might have the same departure and arrival country.

No two flights will have the same arrival time, or have the same departure time. In addition, no flight will have the same arrival time as the departure time of another flight. Finally, it is guaranteed that there will always be a way for David to arrive at his destination.

Output

Print, on a single line, the minimum sum of frustration.

Time limit: solutions should complete in no more than 5 CPU seconds.

Examples

In the first sample, it is optimal to take this sequence of flights:

- Flight 5. Goes from airport 1 to airport 2, departing at time 3, arriving at time 8.
- Flight 3. Goes from airport 2 to airport 1, departing at time 9, arriving at time 12.
- Flight 7. Goes from airport 1 to airport 3, departing at time 13, arriving at time 27.
- Flight 8. Goes from airport 3 to airport 5, deparing at time 28, arriving at time 100.

The frustration for each flight is $3^2, 1^2, 1^2$, and 1^2, respectively. Thus, the total frustration is 12.

Note that there is an itinerary that gets David to his destination faster. However, that itinerary has a higher total frustration.

Example 1

Sample Input

```
5 8
1 2 1 10
2 4 11 16
2 1 9 12
3 5 28 100
1 2 3 8
4 3 20 21
1 3 13 27
3 5 23 24
```

Sample Output

```
12
```

Example 2

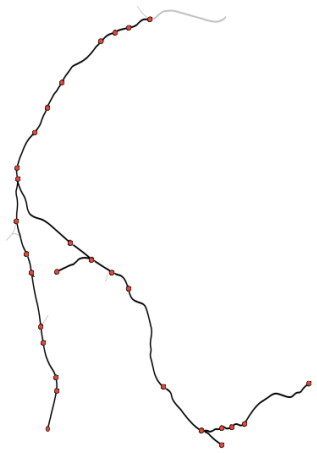
Sample Input

```
3 5
1 1 10 20
1 2 30 40
1 2 50 60
1 2 70 80
2 3 90 95
```

Sample Output

```
1900
```

Problem E: Rainbow Roads



The Transit Authority of Greater Podunk is planning its holiday decorations. They want to create an illuminated display of their light rail map in which each stretch of track between stations can be illuminated in one of several colors.

At periodic intervals, the controlling software will choose two stations at random and illuminate all of the segments connecting those two stations. By design, for any two stations on the Greater Podunk Railway, there is a unique path connecting the two.

For maximum color and cheer, the display designers want to avoid having two adjacent segments of track lighting up in the same color. They fear, however, that they may have lost track of this guideline in the process of building the display. One of them has gone so far as to propose a means of measuring just how far from that ideal they may have fallen.

Description

You are given a tree with n nodes (stations), conveniently numbered from 1 to n. Each edge in this tree has one of n colors. A path in this tree is called a *rainbow* if all adjacent edges in the path have different colors. Also, a node is called *good* if every simple path with that node as one of its endpoints is a *rainbow* path. (A simple path is a path that does not repeat any vertex or edge.)

Find all the *good* nodes in the given tree.

Input

The first line of input contains a single integer n ($1 \leq n \leq 50\,000$).

Each of the next $n - 1$ lines contains three space-separated integers a_i, b_i, and c_i ($1 \leq a_i, b_i, c_i \leq n$; $a_i \neq b_i$), describing an edge of color c_i that connects nodes a_i and b_i.

It is guaranteed that the given edges form a tree.

Output

On the first line of the output, print k, the number of good nodes.

In the next k lines, print the indices of all good nodes in numerical order, one per line.

Time limit: solutions should complete in no more than 3 CPU seconds

Examples

(For the first sample, node 3 is good because all paths that have node 3 as an endpoint are rainbow. In particular, even though the path $3 \to 4 \to 5 \to 6$ has two edges of the same color (i.e. $3 \to 4$, $5 \to 6$), it is still rainbow because these edges are not adjacent.)

Example 1

Sample Input

```
8
1 3 1
2 3 1
3 4 3
4 5 4
5 6 3
6 7 2
6 8 2
```

Sample Output

```
4
3
4
5
6
```

Example 2

Sample Input

```
8
1 2 2
1 3 1
2 4 3
2 7 1
3 5 2
5 6 2
7 8 1
```

Sample Output

```
0
```

Example 3

Sample Input

```
9
1 2 2
1 3 1
1 4 5
1 5 5
2 6 3
3 7 3
4 8 1
5 9 2
```

Sample Output

```
5
1
2
3
6
7
```

Example 4

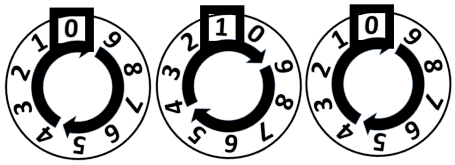
Sample Input

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

Sample Output

```
4
1
6
7
9
```

Problem F: Spinning Up Palindromes



"Sabotage!" exclaimed J.R. Diddly, president and founder of Diddly Widgets Inc.

"Vandalism, perhaps. Nothing's actually been damaged." responded Robert Lackey, the chief accountant.

Both were staring up at the large counter suspended above the factory floor, a counter that had faithfully recorded the number of widgets that had come off the assembly line since the factory was opened. But someone had changed the number being displayed so that it formed...

"It's a palindrome." said Lackey. "It reads the same forwards as backwards."

"What I don't understand," said Diddly, "is why our security guards didn't catch the vandals during their regular sweeps. It must have taken them hours to click forward to this new number, one step at a time."

"No." replied Lackey. "Although we only advance the rightmost digit each time a new widget is built, it's possible to spin any of the digits. With a little planning, this might have taken only a few seconds."

Description

Consider a digital counter consisting of k wheels, each showing a digit from `0` to `9`. Each wheel is mounted so that it can advance to the next digit in a single step, *e.g.*, from `3` to `4`, or from `8` to `9`.

It is also possible to advance from digit `9` to digit `0`. However, when this happens, the wheel on its immediate left will also advance to the next digit automatically. This can have a cascade effect on multiple wheels to the left, but they all happen in a single step.

Given the current setting of the counter, find the smallest number of steps until one can reach a palindrome. The palindrome must respect leading zeros, *e.g.*, `0011` is not a palindrome.

For example, for input `610`, it takes four steps. This can be done by incrementing the `6` wheel four times, resulting in `010`.

Input

Input will consist of single line containing an integer of 1 to 40 digits. The number of digits in the input is the number of wheels on the counter. Numbers may contain leading zeros.

Output

Print a single line of output containing one integer, the minimum number of wheel advances required to produce a palindrome.

Examples

Example 1

Sample Input

```
0
```

Sample Output

```
0
```

Example 2

Sample Input

```
009990001
```

Sample Output

```
3
```

Example 3

Sample Input

```
29998
```

Sample Output

```
5
```

Example 4

Sample Input

```
610
```

Sample Output

```
4
```

Example 5

Sample Input

```
981
```

Sample Output

```
2
```

Example 6

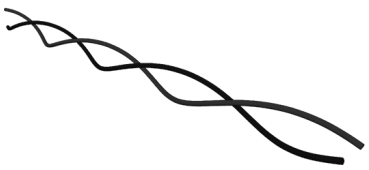
Sample Input

```
908419470094090379719171824780119701926 8
```

Sample Output

```
54
```

Problem G: Long Long Strings



To store DNA sequences your company has developed a LongLongString class that can store strings with up to ten billion characters. The class supports two basic operations:

- `Ins(`p`, `c`)`: Insert the character c at position p.
- `Del(`p`)`: Delete the character at position p.

A DNA editing program is written as a series of `Ins` and `Del` operations. Your job is to write a program that compare two DNA editing programs and determine if they are identical, *i.e.*, when applied to any sufficiently long string, whether the end result is the same.

For example:

- `Del(1) Del(2)` and `Del(3) Del(1)` are identical.
- `Del(2) Del(1)` and `Del(1) Del(2)` are different.
- An empty sequence and `Ins(1, `X`) Del(1)` are identical.
- `Ins(14, `B`) Ins(14, `A`)` and `Ins(14, `A`) Ins(15, `B`)` are identical.
- `Ins(14, `A`) Ins(15, `B`)` and `Ins(14, `B`) Ins(15, `A`)` are different.

Input

Input will consist of the descriptions of two DNA editing programs.

Each program will consist of some number of operations (between 0 and 2 000). Each operation will be given on its own line. The first character of the line will be `D` for a `Del` operation, `I` for an `Ins` operation, or `E` marking the end of the program.

A `Del` operation will have the `D` character, followed by a space, and then a single integer between 1 and 10^{10}, indicating the character position to delete. All characters after this deleted character will be shifted one position lower.

An `Ins` operation will have the `I` character, followed by a space, and then a single integer between 1 and 10^{10}, indicating the location to insert the new character; all pre-existing characters with this index and higher will be shifted one position higher. Following this integer will be another space and then an uppercase alphabetic character that is the character to insert.

Output

If the two programs are identical, print "`0`" on a single line (without quotation marks). Otherwise, print "`1`" on a single line (without quotation marks).

Time limit: solutions should complete in no more than 3 CPU seconds

Examples

Example 1

Sample Input

```
D 1
D 2
E
D 3
D 1
E
```

Sample Output

```
0
```

Example 2

Sample Input

```
D 2
D 1
E
D 1
D 2
E
```

Sample Output

```
1
```

Example 3

Sample Input

```
I 1 X
D 1
E
E
```

Sample Output

```
0
```

Example 4

Sample Input

```
I 14 B
I 14 A
E
I 14 A
I 15 B
E
```

Sample Output

```
0
```

Example 5

Sample Input

```
I 14 A
I 15 B
E
I 14 B
I 15 A
E
```

Sample Output

```
1
```

Problem H: Haiku Formatting

A haiku is a three-line poem in which the first and third lines contain 5 syllables each, and the second line contains 7 syllables.

An example of a haiku is:

Blue Ridge mountain road.
Leaves, glowing in autumn sun,
fall in Virginia.

Write a program to examine a line of English text and and attempt to render it as a haiku. This will require counting the syllables in the words of the text, which should be done according to the following rules:

- A word consists of a non-empty, maximal string of zero or more alphabetic characters (upper and/or lower-case) followed by zero or more non-blank, non-alphabetic characters.
  - Upper/lower case distinctions are ignored for the purpose of counting syllables, but must be retained in the final output.
  - Non-alphabetic characters are ignored for the purpose of counting syllables, but must be retained in the final output.
- The characters 'A', 'E', 'I', 'O', 'U', and 'Y' are vowels. All other alphabetic characters are consonants.

  Exceptions to this rule:
  - The character sequence "QU" is considered to be a single consonant.
  - The letter 'Y' is considered to be a consonant if it is immediately followed by one of the other vowels.
- Every word has at least one syllable.

  For example, "Fly","I", "!?", and "Ssshhh!" are words of one syllable.
- Each (maximal) string of one or more consonants with at least one vowel to either side indicates a division into separate syllables.

  For example, "strong" has one syllable, "stronger" has 2, and "bookkeeper" has 3. "player" has two syllables (because the 'y', being followed by an 'e', is considered a consonant).

  Exceptions to this rule are:
  - An 'E' appearing as the last alphabetic character in a word is silent and should be ignored unless the next-to-last alphabetic character is an 'L' and the character immediately before that is another consonant.

    For example, "cake", "ale" and "pale" have one syllable. "able" has two.
  - An 'ES' sequence at the end of the alphabetic sequence in a word does not add a syllable unless immediately preceded by two or more consonants.

    For example, "ales" and "pales" have one syllable. "witches" and "verses" have two.

Input

Input will consist of a single line of text consisting of a sequence of one or more words (as defined above) separated by single blanks.

The total line length will not exceed 200 characters.

Output

If the words in the input line can be divided into a haiku, then print the haiku as three lines of output.

- Each line should be left-justified.
- A single space should separate each pair of words within a line.
- Upper/lower casing from the input should be preserved.
- Non-alphabetic characters terminating each word should be preserved.
- A word cannot be split across multiple lines.

If the words in the input cannot be divided into a haiku, print the line of input with no changes.

Examples

Example 1

Sample Input

```
Blue Ridge mountain road. Leaves, glowing in autumn sun, fall in Virginia.
```

Sample Output

```
Blue Ridge mountain road.
Leaves, glowing in autumn sun,
fall in Virginia.
```

Example 2

Sample Input

```
Who would know if we had too few syllables?
```

Sample Output

```
Who would know if we had too few syllables?
```

Example 3

Sample Input

```
International contest- motivation high Programmers have fun!.
```

Sample Output

```
International
contest- motivation high
Programmers have fun!.
```

Example 4

Sample Input

```
Programming contest is stressing us all out. International pain.
```

Sample Output

```
Programming contest is stressing us all out. International pain.
```