Finals 2018

sponsored by:

Problem 1: Minecrypt

Sean and Aisling want to chat about Minecraft all the time but their parents have setup an Internet filter that blocks any message that contains words related to the game like *minecraft*, *creeper*, etc. In order to workaround the filter, Aisling and Sean have designed an encryption system that consists on reversing the letters of the words. For example, they would write *repeerc* instead of *creeper* to avoid the restriction.

Write a problem that given 2 words, s and t computes if one if the correct *encryption* of the other.

Input

The first line will contain the word `s` and the second line will contain the word `t`. The words will only contain lowercase letters. The words won't have more than 100 characters.

Output

If the word `t` is the word `s` written in reverse order print `YES`, otherwise print `NO`.

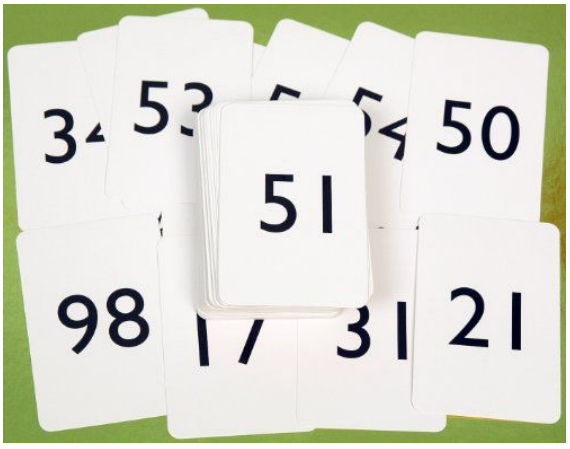
Example 1

Input	Output
`creeper` `repperc`	`YES`

Example 2

Input	Output
`code` `code`	`NO`

Problem 2: Missing Numbers



Sinead loves playing with a deck of 100 cards she has at home, with each card numbered from 0 to 99 and all the cards having different values. When she was at school, her little brother went messing around her room and now Sinead only has N cards.

Can you help her find which cards are missing?

Input

The first line of the input will be an integer N, the number of cards remaining.

The second line will contain N integers from 0 to 99. Each number will appear at most once.

Output

Print the list of the 100 - N missing numbers in ascending order.

Example 1

Input	Output
98 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84	42

85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 ** numbers on second line all on one line*	

Example 2

Input	Output
85 82 94 52 35 72 4 49 41 95 74 85 2 27 99 84 83 26 96 34 65 16 22 92 15 93 89 64 67 21 19 39 0 13 8 33 88 75 28 18 77 61 68 40 45 69 25 5 62 9 36 31 12 20 38 63 6 47 1 53 76 50 98 80 3 42 44 70 59 86 23 29 48 17 37 60 91 54 87 43 7 79 58 97 71 78 ** numbers on second line all on one line*	10 11 14 24 30 32 46 51 55 56 57 66 73 81 90 ** output all on one line*

Problem 3: Cool Subsequence

Given a sequence of lowercase letters, compute the longest contiguous subsequence whose characters are sorted alphabetically.

For example: The sequence `babca` contains several sorted subsequences. The longest of them is `abc`.

Input

A single line with a string of lowercase characters.
For 50% of the test cases, the size of the string will be at most 100.
For the rest, the size of the string will be at most 10000.

Output

Write a single line with the size of the longest contiguous subsequence that is sorted alphabetically.

Examples

Input	Output
`aipo`	`3`
`zvgba`	`1`
`qwertyuiop`	`4` *The subsequence erty is the longest subsequence that is sorted.*

Problem 4: Prime One

A prime number is a positive integer greater than 1 that is only divisible by 1 and the number itself. Numbers such as 7, 13 or 131 are prime numbers while 9, 10 or 144 are not.

Write a program that, given a integer **N**, computes how many numbers less than or equal to N exist such that there are prime numbers *and* the their first digit is 1.

The very first numbers that meet the requirements are: 11, 13, 17, 19, 101, 103 and 107.

Input

The only input will be an integer number **N**.
For 50% of the test cases, the **N** will be less than 2000.
For the rest, **N** will be at most 2000000 (2 million).

Output

Output the amount of numbers that are both primes and their first digit is 1 up to, and including **N**.

Examples

Input example 1	Output example 1
`100`	`4` *`[11, 13, 17, 19]`*
Input example 2	Output example 2
`137`	`12` *`[11, 13, 17, 19, 101, 103, 107, 109, 113, 127, 131, 137]`*

Problem 5: Potholes

The road that connects Sligo and Galway is in really bad shape and has a lot of potholes and broken sections. The government has finally agreed to put up some money to fix it but the budget they promised won't be enough to cover for the repairs of the whole road.

We will represent the road as a string of dots and Xs: The string `.XX..` represents a road of size 5 where the second and third section need repairs and the rest is fine. The road `..XXX...XX..` has 2 sections that are broken.

We have found a company that, for some reason, charges us by continuous segments repaired instead of the distance or time required so we want to take advance of this and repair the road as much as possible. That is, repairing `X` section will cost us the same as repairing `XX` or even `XXXXXXX`.

For example: If we get enough money from the government to repair only one segment of the road `..XXX...XX..` we want to fix the first one because leaves us with only two broken sections afterwards.

Input

The first single line contains the state of the road. A `.` represents a segment that it is good shape. `X` represents a broken segments that need repairs.

The second line contains an integer stating the number of continuous segments that can be repaired.

In 50% of the cases, the length of the road will be at most 1000 characters.
For the rest, the length will be at most 1000000 (1 million)

Output

Print the minimum number of sections that will remain broken after the fixes.

Examples

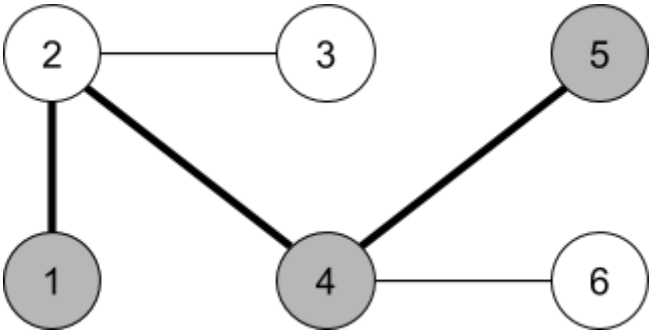
Input	Output
.xx.. 1	0
Input	Output
..xxx...xx.. 1	2

Problem 6: Pothole Profit

As you know from the previous problem, there is a road maintenance company in Ireland with a weird billing schema. The owners saw they were losing a lot of money so they decided to expand their operations to the whole Ireland.

In Ireland there are n cities, connected by n - 1 two-way roads. The cities are numbered from 1 to n. You can get from one city to another moving along the roads. The roads that you need to use to move between two cities form a *path*.

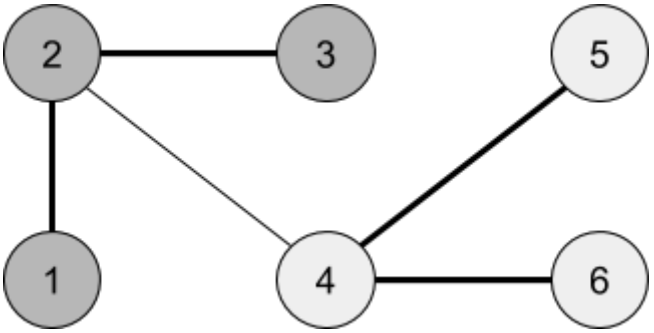
The example below shows a map with 6 cities. In darker colors we marked a path from node 1 to 5. The path has length 3.



The company has won the tender from the Irish Government to repair two paths of roads and they are allowed to choose the paths to repair themselves. The only condition they have to meet, is that the two paths shouldn't cross (i.e. shouldn't have common cities).

The money that the company will charge is the product of the lengths of the two paths. That is, if they repair one path of length 10 and one path of length 5, the cost will be 50 (10 * 5). We will consider that the length of each road is 1, and the length of a path equals the amount of roads in it.

For the example above, if the company repairs the paths 1 - 2 - 3 (length 2) and 5 - 4 - 6 (length 2). They will get a profit of 4. See diagram below.



You task is to find the maximum possible profit for the company depending on the network of roads.

Input

The first line contains an integer n ($2 \le n \le 200$), where n is the amount of cities in the country. The following $n - 1$ lines contain the information about the roads. Each line contains a pair of numbers of the cities, connected by the road a, b ($1 \le a, b \le n$).

Output

Print the maximum possible profit the company can get.

Examples

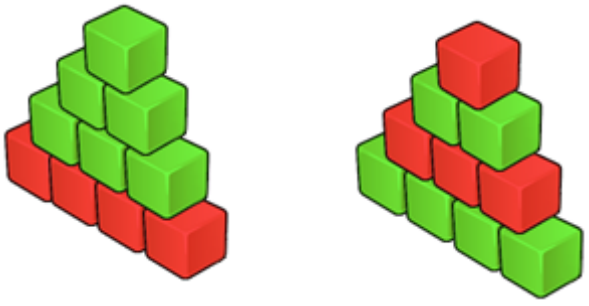
Input	Output
4 1 2 2 3 3 4	1
7 1 2 1 3 1 4 1 5	0

1 6 1 7	
Input 6 1 2 2 3 2 4 5 4 6 4	Output 4 *This is the graph from the problem statement.*

Problem 7: Triangle

We have **r** red and **g** green blocks that we want to use to construction the great red-green triangle. The great red-green triangle can be built following these strict rules:

- The great red-green triangle consist of some number of levels.
- Given red-green triangle with **n** levels, then the first level of this triangle should consist of n blocks, second level, of **n** - 1 blocks, the third one, of **n** - 2 blocks, and so on; the last level of such triangle should have one block. In other words, each successive level should contain one block less than the previous one.
- Each level of the red-green triangle should contain blocks of the same color.



Let **h** be the maximum possible number of levels of red-green triangle, that can be built out of **r** red and **g** green blocks meeting the rules above. The task is to determine how many different red-green triangles having **h** levels can be built out of the available blocks.

Two red-green triangles are considered different if there exists some level, that consists of red blocks in the one triangle and consists of green blocks in the other triangle.

You are to write a program that will find the number of different red-green triangles of height **h** modulo 1000000007 (10^9 + 7).

Input

The only line of input contains two integers **r** and **g**, separated by a single space, the number of available red and green blocks respectively (0 ≤ **r**, **g** ≤ 2*10^5, **r** + **g** ≥ 1).

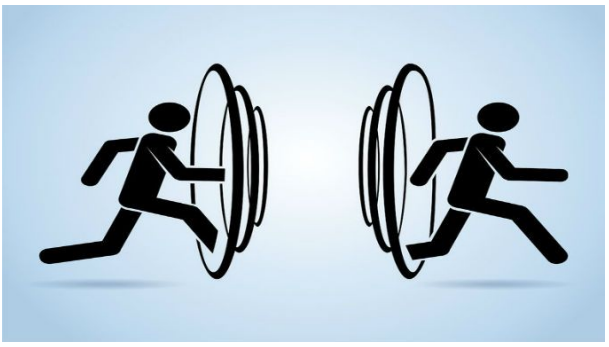
Output

Output the only integer, the number of different possible red-green triangles of height **h** modulo 10^9 + 7.

Examples

Input	Output
4 6	2 *The image in the problem statement shows all possible red-green triangles for the first sample.*
9 7	6
1 1	2

Problem 8: Teleporters



The year is 3018 and you are participating in a DCU Physics competition that involves crossing Ireland from west to east along a straight line segment using teleporting technology. Initially you are located at the westmost point of the segment (Roundstone, County Galway). It is a rule of the competition that you must always move along the segment, and always eastward.

There are **N** teleporters on the segment. A teleporter has two endpoints. Whenever you reach one of the endpoints, the teleporter immediately teleports you to the other endpoint. (Note that, depending on which endpoint of the teleporter you reach, teleportation can transport you either eastward or westward of your current position.) After being teleported, you must continue to move eastward along the segment; you can never avoid a teleporter endpoint that is on your way. There will never be two teleporter endpoints at the same position. Endpoints will be strictly between the start and the end of the segment.

Every time you get teleported, you earn 1 point. The objective of the competition is to earn as many points as possible. In order to maximize the points you earn, you are allowed to add up to **M** new teleporters to the segment before you start your journey. You also earn points for using the new teleporters.

You can set the endpoints of the new teleporters wherever you want (even at non-integer coordinates) as long as they do not occupy a position already occupied by another endpoint. That is, the positions of the endpoints of all teleporters must be unique. Also, endpoints of new teleporters must lie strictly between the start and the end of the segment.

Note that it is guaranteed that no matter how you add the teleporters, you can always reach the end of the segment.

TASK
Write a program that, given the position of the endpoints of the **N** teleporters, and the number **M** of new teleporters that you can add, computes the maximum number of points you can earn.

CONSTRAINTS
1 <= **N** <= 1,000,000
The number of teleporters initially on the segment.
1 <= **M** <= 1,000,000
The maximum number of new teleporters you can add.
1 <= $\mathbf{W_x}$ < $\mathbf{E_x}$ <= 2,000,000
The distances from the beginning of the segment to the western and eastern endpoints of teleporter X.

INPUT
Your program must read from the standard input the following data:

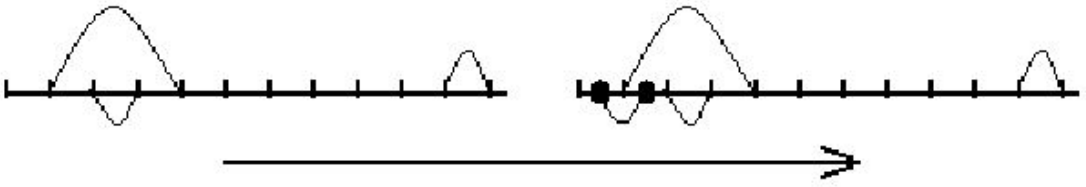
- Line 1 contains the integer **N**, the number of teleporters initially on the segment.
- Line 2 contains the integer **M**, the maximum number of new teleporters that you can add.
- Each of the next **N** lines describes one teleporter. The $\mathbf{i^{th}}$ of these lines describes the $\mathbf{i^{th}}$ teleporter. Each line consists of 2 integers: $\mathbf{W_i}$ and $\mathbf{E_i}$ separated by a space. They represent respectively the distances from the beginning of the segment to the western and eastern endpoints of the teleporter.

No two endpoints of the given teleporters share the same position. The segment that you will be travelling on starts at position 0 and ends at position 2,000,001.

OUTPUT
Your program must write to the standard output a single line containing one integer, the maximum number of points you can earn.

Examples

Input	Output
3 1 10 11 1 4 2 3	6



The first figure shows a segment with the three original teleporters. The second figure shows the same segment after adding a new teleporter with endpoints at 0.5 and at 1.5.

After adding the new teleporter as shown in the figure, your travel would be the following:

- You start at position 0, moving eastward.
- You reach the endpoint at position 0.5 and get teleported to position 1.5 (you earn 1 point).
- You continue to move east and reach endpoint at position 2; you get teleported to position 3 (you have 2 points).
- You reach endpoint at position 4, and get teleported to 1 (you have 3 points).
- You reach endpoint at 1.5, and get teleported to 0.5 (you have 4 points).
- You reach endpoint at 1, and get teleported to 4 (you have 5 points).
- You reach endpoint at 10, and get teleported to 11 (you have 6 points).
- You continue until you reach the end of the segment finishing with a total score of 6 points.

Input	Output
3 3 5 7 6 10 1999999 2000000	12