Python Programming Contest

Computer Systems

Version 1
11-26-2024

1. Candy Gifts

Alice and Bob are preparing gifts for the students of their school. Together they decided that the best (not necessarily for teeth) gift would be lots of candy. Bob bought *N* packages of candy, containing 1, 2, ..., *N* candies in sequence.

Alice wants to distribute all the candies among as many children as possible, but she doesn't like the idea of opening the packages (who would want to get an open gift?) or the idea that some students might get a different number of candies than others. However, one person can get several different packages.

Can you help? Write a program that determines the largest number of students between whom all the packages can be divided so that everyone gets the same amount of candy each.

Input

The first (only) line of the input contains one natural number *N* ($1 \leq N \leq 200\,000$) specifying the number of packages prepared by Bob.

Output

The first line of output should contain one natural number *R*: the largest number of children who can be given gifts. In the following *R* lines, a description of the gifts for the following children should be written out, one per line.

The description of the gifts for successive children should consist of a natural number *Ri* specifying the number of packages the *i*th child will receive, a single space, and a sequence *Ri* of natural numbers describing the numbers of candies in the packages assigned to the gift for the *i*th child.

If there are multiple possible solutions, your program can output any of them.

Examples

Sample Input	**Sample Output**

3	2
	2 1 2
	1 3

4	2
	2 1 4
	2 2 3

Explanation of the example: Bob has prepared packages containing 1, 2, 3, and 4 candies. They can be given to two students: one by giving packages containing 2 and 3 candies, and the other by giving packages containing 1 and 4 candies.

2. Courses Prerequisites

In a bustling city of scholars, a new university has been established, offering a diverse array of courses. The university has a unique structure: each course builds on the knowledge of others, creating a web of prerequisites that every student must navigate.

You, a promising student, are eager to graduate with honors. However, there is one challenge: to plan your journey through the courses so that you meet all prerequisites without skipping any.

The university provides you with a list of courses (of length **n**), labeled from 0 to **n** - 1, and **m** rules in the form of pairs (prerequisites). Each pair [a, b] indicates that course b must be completed before course a.

Your task is to determine a valid order to take the courses so that you can finish all of them. If multiple valid paths exist, any will suffice. However, if it's impossible to meet the prerequisites for all courses, your dream of graduating will shatter, and you must return an empty plan (empty array).

Input

The program will be given two inputs:

- The total number of courses (n), labeled from 0 to n - 1.
- The total number of prerequisite rules (m), followed by m pairs of integers. Each pair [a, b] indicates that course b must be completed before course a.

Output

The program should return:

- A list of integers representing a valid order in which the courses can be taken to satisfy all prerequisites.
  - If multiple valid orders exist, return any of them.
- If it is impossible to complete all courses due to circular dependencies, return an empty list.

Sample Input	**Sample Output**
4 **4** **0 2** **3 1** **2 1** **0 1**	**[1, 2, 0, 3]** (or any valid topological order)
3 **0**	**[0, 1, 2]** (or any valid topological order)
3 **3** **0 1** **1 2** **2 0**	**[]**

3. Factory Fulfilment

In a Factory we have a list of 5 items with their respective dependencies with the intention to be created eventually and other items that could be made from that such item.

After hours having to check the list, you decided to create a program that takes from the list of items and their dependencies and gives a sequence of items in order of requirements such that no item would have a missing requirement previously mentioned in the output list.

While writing the list you notice that there is no item with more than 2 dependencies.

Input

The input must be the item we want to add followed by its dependency (0, 1 or 2) an item per row.

Output

The output must show the sorted list of items by requirements. It should handle errors like cycles or invalid input.

Sample Input	**Sample Output**
Iron_Ore Iron_Plate Iron_Plate Electronic_Circuit Copper_Ore Copper_Plate Copper_Plate Copper_Circuit Electronic_Circuit Electronic_Circuit Assembly_Machine	['Copper_Ore', 'Copper_Plate', 'Copper_Circuit', 'Iron_Ore', 'Iron_Plate', 'Electronic_Circuit', 'Assembly_Machine']
A A	Item 'A' cannot depend on itself
C A C B	Item 'C' already entered
C D B N	Invalid number of dependencies (2 max)
A B C B C A C A B E F G F G E	No valid sequence exists
A B C D E	['E', 'D', 'C', 'B', 'A']

4. Linux Installation

Student helpers' updates Linux on students' computers. Initially, only one computer has installed the latest version of Linux. Unfortunately, the eduroam has broken down, so student helpers can now upload new versions of Linux to other computers only via cables. Student helpers can connect two computers with one cable at a time, and at most one cable can be connected to the computer at a time. Copying files of operating system between computers takes exactly one hour.

Count how many hours it will take for student helpers to install the latest version of Linux on all students' computers, if there are n of them and student helpers brought k cables with them.

Input

The line contains two integers n, k ($1 \le k \le n \le 10^{18}$) - the number of computers and cables.

Output

Your program should write the minimum number of hours needed to implement installing Linux on all computers.

Examples

Sample Input	**Sample Output**
8 3	4
6 6	3
7 1	6
1 1	0
6 2	3

5. Palindromic substrings

You are given a string s. Your task is to find all distinct palindromic substrings that have a length of 2 or more. A palindromic substring is a sequence of characters in the string that reads the same forwards and backwards, such as "aba" or "madam".

Once you've found all the palindromic substrings, you should sort them in alphabetical order (lexicographical order) and print them, each on a new line. If no palindromic substrings of length 2 or more exist, simply print "No Palindromes".

Input

The input should contain a string, which consists of palindromes (or not).

Output

Your program should write all the palindromes that are within the string input and sort them in alphabetical order.

Examples

Sample Input	**Sample Output**
hahaha	aha ahaha hah hahah
alphabetical	No palindromes
commodore64	mm odo ommo
uwu	uwu
computer systems	sys

6. Savings Calculator

A just graduated Software Engineer wants to buy his dream Laptop which costs *x* DKK. He has *y* DKK saved up, but it is not enough to buy the laptop, so he decides to put the money in an investment. Every lucky week his money is increasing with the basic interest rate which is 5%. Every unlucky week the interest rate becomes -3%. Every 4th week is an unlucky week. So the first week he has his savings as a base in the investment, the second week he has the increased amount as a base in the investment. Write a program which counts how many weeks does he need to invest his money to get the dream laptop.

Input

The input line contains 2 integers *x*, *y*, where *x, y* > 0 (x: price of the laptop, y: saved money)

Output

The output writes the number of weeks needed in the investment.

Sample Input	**Sample Output**
100 81	6
1000 1000	0
9876 5432	19
10000 9999	1
18000 7800	27

7. Trains

A train company is experimenting with new trains made up of wagons with varying lengths. However, an issue that they quickly noticed is that they have to be really careful on how they assemble a train from their wagons. If they exceed the length of the train stations not all wagons will be accessible by the passengers. If they create trains that are shorter than the train stations, then passengers might be waiting where no wagon stops. Therefore, the train should exactly match the size of the train stations. Luckily, the train stations were designed so that they are the same length. The designers of the new trains wondered how many different ways they can assemble a train, so they hired you to help them figure it out.

Input

The first row of the input are the lengths of the different types of wagons. You can use any number of each type of wagon. The second row contains the number for how long the train stations are.

Output

A single number describing how many ways you can assemble a train from the wagons to match the length of the train stations.

Sample input	**Sample output**
2 3 10	7
2 4 8 11	0
2 5 10 50 100 680	4160821

8. University Sweepstake

The university set up a sweepstakes for all the currently enrolled students at the Sønderborg campus, with the possibility to win a new iPad.

All the students receive a number assigned to them which is stored in an ordered array. To determine which student won, we start from the left and traverse the array eliminating the first number and every other number in the array. Once arrived at the end of the array, we traverse the array once again, but this time going from the right to the left, again eliminating every other number, starting from the first.

We do this over and over again until only one number remains, which is the winner.

Tip: You don't necessarily need an array

Input

The number of the total students participating in the sweepstake (n > 0)

Output

The student who won the sweepstake

Sample input	**Sample output**
11	8
1238	472
854	344
3	2