.

FRISCO First Bytes High School Programming Contest fall 2018
Advanced Category

A **Fish Tank**

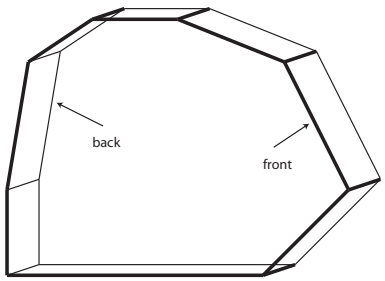
Data File: A.txt
Time allowed = 10 seconds

You just bought a fish tank that has an interesting shape. If you pour L liters of water into the tank, how high will the water be in the tank?

When you look at this tank from the front, or the back, it has the shape of a convex polygon. This polygon has exactly two vertices on the table (y-coordinates are 0), and all other vertices have positive y-coordinates. There are also exactly two vertices with maximum y-coordinates, and water is poured into the opening between these two vertices. The front and back faces of the aquarium are parallel and are D centimeters apart. The tank is glued to the table, so no matter what shape it has, it keeps its position and does not tip over.

All coordinates and lengths in this problem are given in centimeters. It should be noted that each cubic meter is equivalent to 1000 liters.

Below is an illustration showing one configuration of the tank.



Input
The input consists of a single test case. The first line contains an integer $N(4 \leq N \leq 100)$ giving the number of vertices in the polygon. The next line contains two integers D and L, where $1 \leq D \leq 1000$ is the depth of the aquarium tank and $0 \leq L \leq 2000$ is the number of liters of water to pour into the tank. The next N lines each contains two integers, giving the (x, y) coordinates of the vertices of the convex polygon in counterclockwise order. The absolute values of x and y are at most 1000. You may assume that the tank has a positive capacity, and you never pour more water than the tank can hold.

Output
Print the height of the water (in centimeters) in the fish tank on a line to 2 decimal places.

Sample Input 1	Sample Output 1
4 30 50 20 0 100 0 100 40 20 40	20.83

Sample Input 2	Sample Output 2
9 30 70 110 70 100 80 80 80 -10 60 -40 30 -40 25 20 0 100 0 120 10	19.74

B **Order Up!**

Data file = B.txt
Time limit = 10 seconds

You are given up to 500 lists of names. Each list contains between 1 and 200 names. These are the last names of someone's favorite stars of science and math. You need to sort the names into order, but you don't need them sorted exactly. Sort the names into increasing (ASCII) order based on the first two characters of the names.

However, if two names have the same first two characters, they should stay in the same relative order as in the input (this is known as a 'stable sort'). Sorting is case sensitive (uppercase letters come before lower case letters in ASCII, i.e., $A < B < \ldots < Z < a < b < \ldots < z$).

Write a program that sorts a list of last names, where the sort uses the first two letters of the name. Names sharing the same first two letters (case sensitive) should retain the same relative order before and after sorting.

Input:
Input consists of a sequence of K ($1 \leq K \leq 500$) test cases. Each of the K cases starts with a line containing an integer N ($1 \leq N \leq 200$). After this comes N names, made up of only letters (A-Z, lowercase and uppercase), one name per line. Names have between 2 and 20 letters. After K test cases, the end of the input will be indicated by a single integer zero on its own line.

Output:
For each case, print the last names sorted as described above, one name per line. Print a blank line between the test cases.

Sample Input	Output for Sample Input
3	Bach
Mozart	Beethoven
Beethoven	Mozart
Bach	
5	Godel
Hilbert	Hilbert
Godel	Poincare
Poincare	Pochhammer
Ramanujan	Ramanujan
Pochhammer	
0	

C Don't Ever, Ever Gamble

Input File: C.txt
Run Time Limit: 10 sec

In this problem you'll be considering the outcomes of rolling two dice with a variety of numbers of faces.

Task:
Write a program to compute the most likely outcomes for the sum of two dice rolls. Assume each die has numbered faces starting at 1 and that each face has equal roll probability.

Input:
The input consists of up to 20 test cases. Each test case occupies a single line with two space-separated integers, N, M, $4 \leq N, M \leq 20$, specifying the number of faces of the two dice. Input ends with End Of File.

Output:
For each test case, output a line beginning with the test case number on its own line, as illustrated below, followed by the most likely outcome for the sum. In case there are several outcomes with the same probability, they must be listed from lowest to highest value on separate lines.

Sample Input 1	Output for Sample Input 1
6 6 6 4 12 20	Test Case 1: 7 Test Case 2: 5 6 7 Test Case 3: 13 14 15 16 17 18 19 20 21

D **Smallest Multiple**

Data File: D.txt
Time allowed = 10 seconds

Given a list of integers, each of which is in $[1, 2^{32}]$, find the smallest integer that is a multiple of each of the given numbers.

Input:
Input consists of up to 100 test cases, one per line. Each test case contains at least one but no more than 100 space-separated integers, each in the range $[1, 2^{32}]$. Input ends at end of file.

Output:
For each test case, print a line with the smallest positive integer that is a multiple of every number in the set.

Sample Input 1	Output for Sample Input 1
2 3 5	30
1 2 3 4	12
399 772 163 959 242	832307365428

E **Prime Hopping**

Input File: E.txt
Run Time Limit: 10 sec

Every year all the frogs in the forest gather along the big central road to show the other animals their unique ability. They are experts in forming towers by climbing on top of each other. The frogs have arrived at different locations along the central road. They are notorious show-offs: each one always jumping as far as it can and the distance it jumps is always a prime number. Not every frog is as strong as the others, so jumping distances may vary between frogs. Naturally, the frogs only jump to increase their position along the road, never the other way!

The frog king wants to invite all visitors to marvel at the most spectacular frog tower. Multiple frog towers can be created, but the king wants to show the crowd the largest tower at the smallest possible position. He doesn't want anyone to miss the action because they were at the wrong spot! Can you help the frog king determine the position and size of the tower?

Input:
On the first line is one integer n , the number of frogs taking part in the proceedings, $1 \leq n \leq 40$. Then follows n lines with integers x_i and d_i, the initial position and prime jumping distance of the ith frog. Here $0 \leq x_i \leq 2^{60}$ and $2 \leq d_i \leq 10^8$.

It is given that the product of all unique jumping distances is less than 10^9.

Output:
Output a single line with two integers indicating: the smallest position of the highest frog tower, the size of the highest frog tower. Separate these integers by a space. There will always be a unique answer.

Sample Input 1	Output for Sample Input 1
3 0 2 1 2 3 3	3 2

Sample Input 2	Output for Sample Input 2
5 0 2 1 3 3 3 7 5 9 5	12 3

Sample Input 3	Output for Sample Input 3
2 9972 9973 9966 9967	99400890 2

F. **Solitaire Mark X**

Input File: F.txt
Run Time Limit: 10 sec

In this problem you are given a board with 23 cavities located along a line. Some of the cavities are occupied by pebbles. The aim of the game is to remove as many pebbles as possible from the board. Pebbles disappear from the board as a result of **a move**.

A move is possible if there is a straight line of three adjacent cavities, call them A, B, and C, with B in the middle. A is vacant, but B and C each contain a pebble. The move consists of moving the pebble from C to A, and removing the pebble in B from the board. You continue to make moves until no more moves are possible.

The following diagram illustrates a board with only ten cavities and six pebbles. Your board will have 23 cavities and up to 23 pebbles.

Input:
The input begins with a positive integer $1 \leq n \leq 20$ on a line of its own. Thereafter n different game descriptions follow. Each game description consists of one line of input with exactly 23 characters, corresponding to the 23 cavities of the board. Each character is either a '-' or an 'o'. A '-' character denotes an empty cavity, whereas an 'o' character denotes a cavity with a pebble in it. In each game there is at least one pebble.

Output:
For each of the n games in the input, output the minimum number of pebbles left on the board, on a line of its own.

Sample Input	Output for Sample Input
5	2
---oo----------oo------	4
-o--o-oo-----o--o-oo---	6
-o----ooo----o----ooo--	23
oooooooooooooooooooooo	4
oooooooooo-ooooooooooo-	

G Mail Train

Input File: G.txt
Runtime Limit: 10 seconds

A mail train visits N $(1 \leq N \leq 100)$ stations, traveling in one direction only, and stopping at each station. At each station there are packages to be picked up and delivered to other stations. Each package is labeled with its ID number, its weight and the station number to which it must be delivered. Stations are numbered $1, 2, 3, \cdots, N$. The train begins at station 1 and ends its journey at station N.

We wish to maximize the number of packages that are delivered to their correct destinations. In doing so, it is possible that a package may not be picked up, or it may be delivered to the correct destination, or it may be dropped off at some station in between the two. Your job is to write a program that reads in the number of stations, the lists of packages to be picked up at each station, and the train's weight limit, and outputs the maximum number of packages that can be delivered.

Input:
There is just one dataset. It begins with a line containing the integer L giving the weight limit of the train. Then, on a line by itself, is an integer N, giving the number of stations, followed by, for each station, a list of packages as follows:

On a line by itself is an integer Q $(0 \leq Q \leq 100)$ giving the number of packages, followed by Q lines, one for each package, containing three integers separated by spaces as follows: an integer I giving the box ID number, then an integer D giving the station number of the destination address, and then an integer W giving the weight of that package.

Output:
Output an integer on a line by itself giving the maximum number of packages that can be delivered. Do not count the packages that begin at their intended destination.

Sample Input	Output for Sample Input
10 5 3 1 3 2 2 2 1 3 4 6 2 5 5 8 4 5 2 2 8 4 3 7 4 1 3 9 4 2 10 5 20 6 5 1 2 12 1 5 13 4 4	6