Staff Student Competition 2018

25th April 2018

Rules and Instructions

- One computer per team.
- You may bring any printed material, including books.
- Use of electronic material is not allowed, except for a language reference on your computer.
- Use of the internet to access any sites other than the competition is not allowed.
- You can print out your code by going to https://mobileprint.manchester.ac.uk/.

A. Building Capacity

Time Limit: 3 Seconds

Problem description

The university has invented a new system! It will be recording a timestamp whenever someone enters and exits a building, using their out-of-hours pass. These times are recorded as $[a_i, b_i]$, where a_i represents when the i^{th} person entered the building, and b_i when that person exited. Thus, the i^{th} person will be considered to be *in* the building at all times t such that $a_i \leq t \leq b_i$. The university administration wants to know the maximum number of people in the building at any one time. Help them figure this out.

Input

The input starts with a single number n ($1 \leq n \leq 10^5$), followed by n lines. Each line contains two space-separated integers, a_i and b_i ($1 \leq a_i, b_i \leq 10^{18}$), the entrance and exit times of the i^{th} person.

Output

Output a single number: the maximum number of people who are in the building simultaneously.

Sample input/output

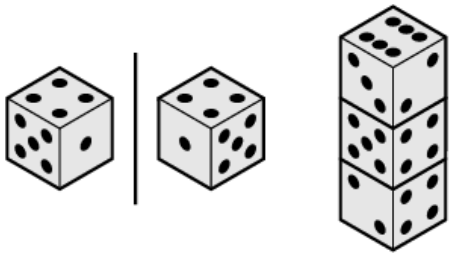
Input 1	Output 1
3 1 2 3 4 5 6	1
Input 2	**Output 2**
3 1 10 2 4 10 12	2
Input 3	**Output 3**
4 1 100 2 50 20 80 101 102	3

B. Dice

Time Limit: 2 Seconds

Problem description

Dice are labelled so that the dots on their opposing faces sum to seven. Therefore, if you can see a corner and its three adjacent faces, you can infer the position of each number on the die.



Alice has built a tower from n dice, constructed in a way so that any two faces that are touching have a different number of dots. Bob wonders if he can uniquely identify the configuration of the dice just by looking at them from the side so that he cannot see all of the faces of the dice (like you see in the figure above). Help Bob tell whether it is possible to uniquely identify the numbers on the faces of all the dice in the tower.

Input

The input starts with a single number n ($1 \leq n \leq 100$), the number of dice in the tower. The second line contains a single number x ($1 \leq x \leq 6$), the number of dots Bob sees on the top of the top die of the tower. Next is n lines each of which contain two space-separated integers. On the ith line two numbers a_i and b_i ($1 \leq a_i, b_i \leq 6; a_i \neq b_i$), the numbers Bob sees on the two sidelong faces of the ith die in the tower, with the dice numbered starting at 1 at the top, and the nth die at the bottom.

Output

Output `YES` if it is possible to to uniquely identify the numbers on the faces of all the dice in the tower. Output `NO` otherwise.

Sample input/output

Input 1	Output 1
3 6 3 2 5 4 2 4	YES
Input 2	**Output 2**
3 3 2 6 4 1 5 3	NO

C. I scream "Ice cream!"

Time Limit: 40 Seconds

Problem description

Valerie Grind has just opened an ice cream parlour. She offers m flavours of ice cream, appetizingly called $F_1, F_2, \ldots, F_m$. On the first day of operation, n small boys, named $B_1, B_2, \ldots B_n$, come in to eat Valerie's ice cream. Each boy will eat as many portions of ice cream as Valerie can give him—including multiple portions of the same flavour—up to a certain number (determined by that boy's stomach capacity), but only those flavours that he likes. In addition, Valerie has limited supplies of each flavour. Determine the maximum amount of ice cream Valerie can sell.

Input

The first line of input contains a space-separated pair of positive integers m and n, the number of flavours and the number of boys, respectively, where $m \leq 100$ and $n \leq 100$. The second line contains a space-separated list of n positive integers, $c_1, \ldots, c_n$, where $c_i \leq 100$ is the total number of portions of ice cream boy B_i can eat ($1 \leq i \leq n$). The third line contains a space-separated list of m positive integers, $p_1, \ldots, p_m$, where $p_j \leq 100$ is the number of portions of flavour F_j which Valerie has to sell ($1 \leq j \leq m$). There follows a sequence of n lines, each consisting of a non-empty, space-separated list of distinct integers in the range $[1, m]$. The ith line in this sequence ($1 \leq i \leq n$) is a list (order unspecified) of all and only the numbers j such that boy B_i likes flavour F_j.

Output

Output a single integer, the maximum number of portions of ice cream that Valerie can sell.

Sample input/output

Input 1	Output 1
2 2 1 3 3 1 1 1 2	4
Input 2	**Output 2**
2 3 1 2 3 3 3 1 2 1 2 1	6
Input 3	**Output 3**
2 3 1 2 4 3 4 1 2 1 2 1	6

D. Path

Time Limit: 1 Second

Problem description

Bill's construction company just recieved a shipment of n square tiles. Bill wants to build the longest path he can by laying them out in full rows of w tiles. Tell Bill how many tiles will be left over when the path is finished.

Input

The input is a single line consisting of two space-separated integers, n ($1 \leq n \leq 10^{18}$), the number of tiles that Bill recieved, and w ($1 \leq w \leq 10^{18}$), the width of the path.

Output

Output a single number: the number of tiles left.

Sample input/output

Input 1	**Output 1**
`10 5`	`0`
Input 2	**Output 2**
`111 7`	`6`
Input 3	**Output 3**
`27 4`	`3`

E. Trouble with Wibbles

Time Limit: 1 Second

Problem description

Wibbles are small, furry creatures of which a great number live on the planet Wibbleheim. They reproduce asexually. Each wibble becomes mature when it is exactly 1 second old. It produces its first young wibble (wiblet) when it is exactly 2 seconds old, and thereafter produces one wiblet every second on the second. The first wibble arrived on Wibbleheim as a newly-mature wibble (exactly 1 second old), left by a passing spaceship.

Wibbles never die, never stop reproducing, and have no predators on Wibbleheim. Thus, the wibble population soon became a matter of concern. In fact, this number is now so huge that the Galactic Administration is only interested in how many *digits* it contains (in base ten).

Write a program to help the Galactic Administration calculate the number of seconds (after the arrival of the first wibble on Wibbleheim) for the wibble population of that planet to reach a given number of digits.

Input

The input consists of a single line, containing a single positive integer, d ($1 \leq d < 2^{32}/5$), the number of digits in the wibble population of Wibbleheim (including any wiblets) when written in standard decimal notation.

Output

The output consists of a single line containing a single non-negative integer t ($0 \leq t < 2^{32}$), the earliest time (measured in seconds from the landing of the first wibble on Wibbleheim) at which the wibble population of the planet, including wiblets, has at least d digits.

Sample input/output

Input 1	**Output 1**
2	5
Input 2	**Output 2**
3	10
Input 3	**Output 3**
100	474

F. Fractal

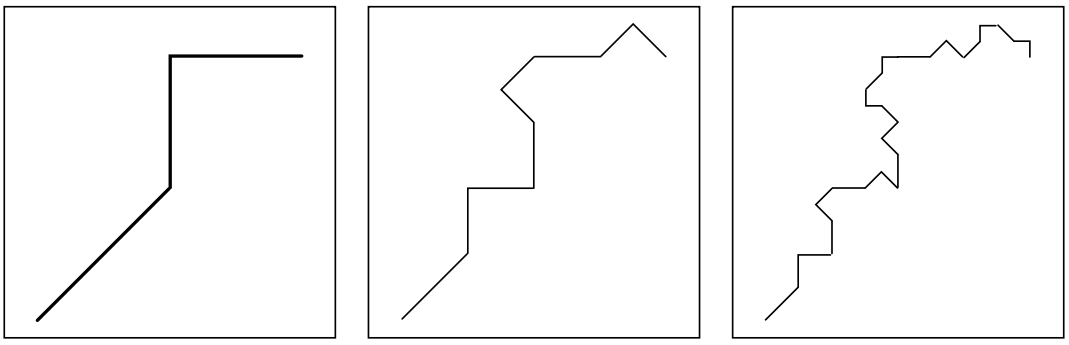
Time Limit: 3 Seconds

Problem description

Fractals are really cool mathematical objects. They have a lot of interesting properties, often including:

- fine structure at arbitrarily small scales;
- self-similarity, i.e., magnified it looks like a copy of itself;
- a simple, recursive definition.

Approximate fractals are found a lot in nature, for example, in structures such as clouds, snowflakes, mountain ranges, and river networks.



In this problem, we consider fractals generated by the following algorithm: we start with a polyline, i.e., a set of connected line segments. This is what we call a fractal of depth one (see leftmost picture). To obtain a fractal of depth two, we replace each line segment with a scaled and rotated version of the original polyline (see middle picture). By repetitively replacing the line segments with the polyline, we obtain fractals of arbitrary depth and very fine structures arise. The rightmost picture shows a fractal of depth three. The complexity of an approximate fractal increases quickly as its depth increases. We want to know where we end up after traversing a certain fraction of its length.

Input

The input starts with a single number n ($3 \leq n \leq 100$), the number of points of the polyline. Then follow n lines with on the ith line two integers x_i and y_i ($-1000 \leq x_i, y_i \leq 1000$), the consecutive points of the polyline. Next follows one line with an integer d ($1 \leq d \leq 10$), the depth of the fractal. Finally, there is one line with a floating point number f ($0 \leq f \leq 1$), the fraction of the length that is traversed. The length of each

line segment of the polyline is smaller than the distance between the first point (x_1, y_1) and the last point (x_n, y_n) of the polyline. The length of the complete polyline is smaller than twice this distance.

Output

The output contains two lines containing x and y respectively, the floating point coordianates of where we end up. The absolute error in both coordinates should be smaller than 10^{-3}.

Sample input/output

Input 1	Output 1
4 -2 -2 0 0 0 2 2 2 3 0.75	0.426777 2
Input 2	**Output 2**
5 0 0 10 0 15 10 20 0 30 0 10 0.3849	10.5472 6.74764
Input 3	**Output 3**
5 0 0 10 0 10 10 20 10 20 20 1 0.5	10 10

G. Virus

Time Limit: 5 Seconds

Problem description

You are given a tree with n nodes. In node 1, which is always the root of the tree, there is a virus that's about to spread to all the other nodes it can reach by traversing the edges. Fortunately, you have time to remove some of the edges before the virus starts to spread. However, only some of the edges can be removed.

Input

The input starts with a single line that contains two space separated integers n and k ($1 \leq k, n \leq 10^5$). Then follow $n-1$ lines. The i^{th} line contains three space-separated integers, a, b and c. This triple indicates that there is an edge between the nodes a and b, with c equal to 1 if that edge can be removed and 0 otherwise.

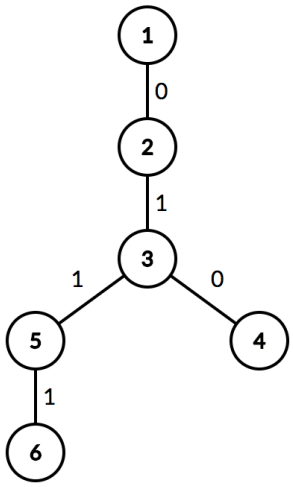
Output

Output the minimum number of edges you need to cut such that the virus will be present in at most k nodes after spreading (including node 1). Otherwise, output -1.

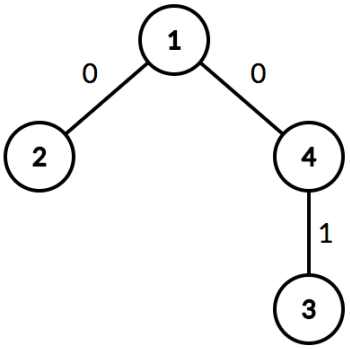
Sample input/output

Input 1	**Output 1**
6 4 3 5 1 1 2 0 2 3 1 3 4 0 5 6 1	1
Input 2	**Output 2**
4 2 1 4 0 1 2 0 3 4 1	-1

Explanation



Sample Input 1: We only cut the edge (3, 5)



Sample Input 2: We are able to cut the edge (3, 4), but the virus can still spread to 3 nodes

H. Banknotes

Time Limit: 1 Second

Problem description

You have an unlimited supply of banknotes with face values of either £a or £b. You want to pay a total of £s using exactly n banknotes.

Input

The input contains only one line with four space-separated integers, a, b, s, n ($1 \leq a, b, s, n \leq 10^9; a \neq b$).

Output

Output the number of banknotes worth £a you need. If there is no solution output -1.

Sample input/output

Input 1	**Output 1**
`1 2 7 5`	`3`
Input 2	**Output 2**
`2 1 7 5`	`2`
Input 3	**Output 3**
`2 1 4 5`	`-1`
Input 4	**Output 4**
`2 3 20 8`	`4`

I. Islands

Time Limit: 5 Seconds

Problem description

The government of an archipelago of several small islands wants to connect them together with bridges so that it is possible to drive from any island to any other island. The administration has outlined a set of pairs of islands that bridges can be built between to achieve this, and have given you coordinates of the islands. Help the government cut costs by minimising the sum of the lengths of the bridges.

For the purposes of calculation, islands may be regarded as points on a flat plane, and the length of a bridge connecting two islands is simply the Euclidean distance between the two islands.

Input

The input starts with a single number n ($1 \leq n \leq 1000$), the number of islands, followed by n lines of two integers, x_i, y_i ($-1000 \leq x_i, y_i \leq 1000, 1 \leq i \leq n$), the coordinates for island i.

After this, you are given an integer m ($1 \leq m \leq \frac{n(n-1)}{2}$), followed by m lines of two integers, i, j ($i \leq 1 \leq n$), the pairs of islands between which it is possible to build a bridge.

Output

Output a single number: the minimum sum of bridge lengths required to connect all the islands. The absolute error should be smaller than 10^{-1}. If it is not possible to connect them all, output "impossible".

Sample input/output

Input 1	Output 1
4 0 0 0 1 1 0 1 1 3 1 2 3 4 1 4	3.41421
Input 2	**Output 2**
4 0 0 1 0 0 1 1 1 2 1 2 3 4	impossible
Input 3	**Output 3**
4 0 0 0 1 10 1 10 10 5 1 3 2 3 3 4 1 4 2 4	29.0499

J. Busy Night

Time Limit: 5 Seconds

Problem description

It's a busy night at the restaurant, and there are n customers waiting to eat. There are m different dishes on the menu and each dish is the same price. Each customer has specified the maximum number of dishes he can afford, but has no preferences as to which dishes he gets, subject only to the constraint that he gets at most one portion of any particular dish. Moreover, there is only a finite number of portions of each dish available. You (the restaurateur) would like to sell as much food as possible.

Input

The input contains on the first line the numbers n ($1 \leq n \leq 10^6$), the number of customers waiting to eat, and m ($1 \leq m \leq 10^6$), the number of different dishes on the menu. The second line contains n space-separated integers, $D_1 \ldots D_n$, ($0 \leq D_i \leq 10^6$) where D_i is the number of dishes the ith person can afford. The third line contains m space-separated integers, $P_1 \ldots P_m$, ($0 \leq P_i \leq 10^6$) where P_i is the number of portions available for the ith dish.

Output

Output one integer: the maximum number of dishes you can sell today.

Sample input/output

Input 1	Output 1
3 4 6 1 4 3 3 3 1	8
Input 2	**Output 2**
2 2 5 2 4 3	4
Input 3	**Output 3**
6 4 11 8 4 9 6 3 11 11 5 1	18