Northwestern European Regional Contest 2014

NWERC 2014

Linköping, November 30

Problems

A Around the Track
B Biking Duck
C Cent Savings
D Digi Comp II
E Euclidean TSP
F Finding Lines
G Gathering
H Hyacinth
I Indoorienteering
J Judging Troubles
K Knapsack Collection

Do not open before the contest has started.

This page is intentionally left (almost) blank.

Problem A
Around the Track
Time limit: 2 seconds

In order to compare race tracks, we wish to compute their lengths. A racetrack is strictly two-dimensional (no elevation). It is described by two simple polygons, where one is completely contained inside the other. The track is the region between these two polygons. We define the length of the track as the absolute minimum distance that one needs to travel in order to complete a lap. This could involve traveling on the very edge of the track and arbitrarily sharp cornering (see Figure A.1).

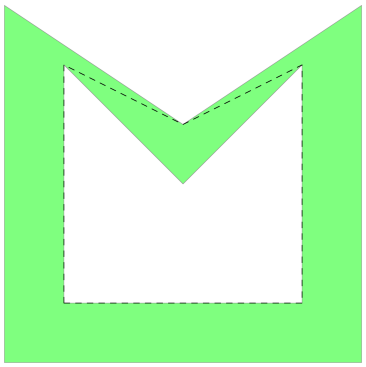


Figure A.1: Illustration of sample input number 3 together with the shortest route around the track (dashed).

Input

The input consists of:

- one line with one integer n ($3 \leq n \leq 50$), the number of vertices of the inner polygon;
- n lines, the ith of which contains two integers x_i and y_i ($-5\,000 \leq x_i, y_i \leq 5\,000$): the coordinates of the ith vertex of the inner polygon;
- one line with one integer m ($3 \leq m \leq 50$), the number of vertices of the outer polygon;
- m lines, the ith of which contains two integers x_i and y_i ($-5\,000 \leq x_i, y_i \leq 5\,000$): the coordinates of the ith vertex of the outer polygon.

For both polygons, the vertices are given in counterclockwise order. The borders of the two polygons do not intersect or touch each other.

Output

Output one line with one floating point number: the length of the race track. Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1

```
3
1 1
2 1
1 2
3
0 0
4 0
0 4
```

Sample Output 1

```
3.41421356237309
```

Sample Input 2

```
5
1 1
5 1
5 5
3 3
1 5
4
0 0
6 0
6 6
0 6
```

Sample Output 2

```
16
```

Sample Input 3

```
5
1 1
5 1
5 5
3 3
1 5
5
0 0
6 0
6 6
3 4
0 6
```

Sample Output 3

```
16.4721359549996
```

Problem B
Biking Duck
Time limit: 2 seconds

Gladstone Gander is walking through Duckburg and needs to get to his date with Daisy Duck as soon as possible. If he doesn't get there in time, Donald might show up and take his place instead.

Picture by D J Shin via Wikimedia Commons, cc by-sa

Duckburg has recently started providing a very eco-friendly way of public transport: bikes. At many bike stations throughout the city, one can pick up a free bike, ride it to another bike station, and drop it there. This gives Gladstone two ways of transportion: on foot or by bike. Biking is faster, of course, but he must pick up and leave the bikes at the designated stations. Gladstone can walk or bike between any two points in a straight line.

Gladstone possesses a map of the (rectangular) center of Duckburg. His current position is on this map and so is the meeting point with Daisy. The map also contains the locations of all bike stations within the boundaries of the map.

There can be way more bike stations though, that are not within the boundaries of the map. Considering his luck, you can assume that the moment Gladstone walks (or bikes) off the map, he encounters a bike station if that suits him well. The bike stations not on the map can be located anywhere outside the map, they do not have to lie on integer coordinates.

That leaves Gladstone with the task of figuring out which route to take. Can you help him out? Given the map and his infinite amount of luck, what is the fastest time to his date with Daisy?

Input

The input consists of:

- one line with two integers v_{walk} and v_{bike} ($1 \leq v_{\text{walk}} < v_{\text{bike}} \leq 1\,000$), the speeds of walking and of biking;
- one line with four integers x_1, y_1, x_2 and y_2 ($-10^6 \leq x_1 < x_2 \leq 10^6$; $-10^6 \leq y_1 < y_2 \leq 10^6$), the bounding coordinates of the map of the center of Duckburg;
- one line with two integers x_{G} and y_{G}, Gladstone's position;
- one line with two integers x_{D} and y_{D}, Daisy's position;
- one line with one integer n ($0 \leq n \leq 1\,000$), the number of known bike stations;
- n lines with two integers x_{station} and y_{station} each, the coordinates of the known bike stations.

All coordinates are on the map of the center, i.e., $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$.

Output

Output one line with the shortest possible time for Gladstone to get to Daisy. Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1

```
1 8
0 0 10 10
5 1
5 9
3
5 8
2 2
9 6
```

Sample Output 1

```
3.000000000
```

Sample Input 2

```
5 100
0 -100000 100000 0
5 -30000
40000 -5
0
```

Sample Output 2

```
501.9987496
```

Problem C
Cent Savings
Time limit: 5 seconds

To host a regional contest like NWERC a lot of preparation is necessary: organizing rooms and computers, making a good problem set, inviting contestants, designing T-shirts, booking hotel rooms and so on. I am responsible for going shopping in the supermarket.

When I get to the cash register, I put all my n items on the conveyor belt and wait until all the other customers in the queue in front of me are served. While waiting, I realize that this supermarket recently started to round the total price of a purchase to the nearest multiple of 10 cents (with 5 cents being rounded upwards). For example, 94 cents are rounded to 90 cents, while 95 are rounded to 100.

Picture by Tijmen Stam via Wikimedia Commons, cc by-sa

It is possible to divide my purchase into groups and to pay for the parts separately. I managed to find d dividers to divide my purchase in up to $d + 1$ groups. I wonder where to place the dividers to minimize the total cost of my purchase. As I am running out of time, I do not want to rearrange items on the belt.

Input

The input consists of:

- one line with two integers n ($1 \leq n \leq 2\,000$) and d ($1 \leq d \leq 20$), the number of items and the number of available dividers;
- one line with n integers $p_1, \ldots p_n$ ($1 \leq p_i \leq 10\,000$ for $1 \leq i \leq n$), the prices of the items in cents. The prices are given in the same order as the items appear on the belt.

Output

Output the minimum amount of money needed to buy all the items, using up to d dividers.

Sample Input 1	Sample Output 1
5 1 13 21 55 60 42	190

Sample Input 2	Sample Output 2
5 2 1 1 1 1 1	0

This page is intentionally left (almost) blank.

Problem D
Digi Comp II
Time limit: 7 seconds

The Digi Comp II is a machine where balls enter from the top and find their way to the bottom via a certain circuit defined by switches. Whenever a ball falls on a switch it either goes to the left or to the right depending on the state of the switch and flips this state in the process. Abstractly it can be modelled by a directed graph with a vertex of outdegree 2 for each switch and in addition a designated end vertex of outdegree 0. One of the switch vertices is the start vertex, it has indegree 0. Each switch vertex has an internal state (L/R). A ball starts at the start vertex and follows a path down to the end vertex, where at each switch vertex it will pick the left or right outgoing edge based on the internal state of the switch vertex. The internal state of a vertex is flipped after a ball passes through. A ball always goes down and therefore cannot get into a loop.

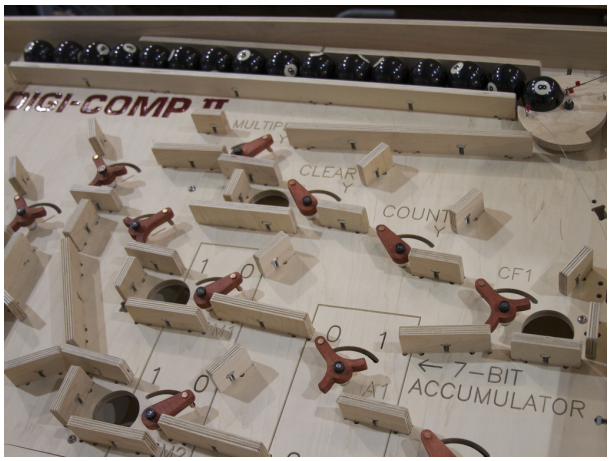


Photo by oskay from Flickr, cc by-sa

One can "program" this machine by specifying the graph structure, the initial states of each switch vertex and the number of balls that enter. The result of the computation is the state of the switches at the end of the computation. Interestingly one can program quite sophisticated algorithms such as addition, multiplication, division and even the stable marriage problem. However, it is not Turing complete.

Input

The input consists of:

- one line with two integers n ($0 \leq n \leq 10^{18}$) and m ($1 \leq m \leq 500\,000$), the number of balls and the number of switches of the graph;
- m lines describing switches 1 to m in order. Each line consists of a single character c ('`L`' or '`R`') and two integers L and R ($0 \leq L, R \leq m$), describing the initial state (c) of the switch and the destination vertex of the left (L) and right (R) outgoing edges. L and R can be equal.

Vertex number 0 is the end vertex and vertex 1 is the start vertex. There are no loops in the graph, i.e., after going through a switch a ball can never return to it.

Output

Output one line with a string of length m consisting of the characters '`L`' and '`R`', describing the final state of the switches (1 to m in order).

Sample Input 1	Sample Output 1
5 3 L 2 3 R 0 3 L 0 0	RLL

Problem E
Euclidean TSP
Time limit: 1 second

The famous Arora-Mitchell approximation algorithm for the Euclidean Travelling Salesman Problem (Euclidean TSP) was discovered independently by Sanjeev Arora and Joseph S. B. Mitchell in 1998. It can approximate the value of an optimal TSP tour in d dimensions within a factor of $1 + 1/c$ in running time

$$n(\log n)^{O((c\sqrt{d})^{d-1})},$$

where n is the number of nodes in the tour.

Photo by psiaki

Miroslava works for a computer security company and it is time to renew a shared cryptographic key in many data centres across Europe. To do this, Miroslava is going to rent a private jet and deliver the key to employees waiting at all major European airports. She wants to be back as soon as possible.

Miroslava's company has a computer that is able to execute p billions of operations per second. Since we can approximate Europe by a two-dimensional plane, we assume that the Arora-Mitchell algorithm runs for exactly

$$\frac{n(\log_2 n)^{c\sqrt{2}}}{p \cdot 10^9}$$

seconds on this computer to produce the exact $(1 + 1/c)$-approximation of the optimal tour.

Miroslava noticed that c is a parameter of the algorithm that can be used to her advantage, but one also needs to be very careful when choosing the right value. If she sets c too low, the algorithm will finish very fast but the time she spends flying around Europe will be too long. On the other hand, setting it too high will force her to wait for an answer from the computer, while she could be flying instead.

Miroslava used to work in a different company and from there she knows that the optimal tour of all major European airports is s meters long, but she wasn't ranked high enough in the company to know the actual tour. Given the speed v of the private jet in meters per second, Miroslava needs $s(1 + 1/c)/v$ seconds to complete the tour produced by the algorithm run with parameter c. For the sake of simplicity, we assume that Miroslava can land, leave a copy of the private key and take off from each airport in an instant.

How long does it take Miroslava to first run the algorithm and then distribute all the keys, assuming that she chooses the optimal parameter c?

Input

The input consists of one line with four numbers:

- an integer n ($4 \leq n \leq 1\,000\,000$), the number of airports;
- a real number p ($0.001 \leq p \leq 5\,000$), the number of billions of operations the computer can execute per second;
- a real number s ($10^6 \leq s \leq 10^9$), the length of the optimal tour of all European airports in meters;

- a real number v ($50 \leq v \leq 900$), the speed of the private jet in meters per second.

All real numbers will have at most 10 digits after the decimal point.

Output

Output one line with the shortest possible time t in seconds to distribute the keys and the value of the parameter c Miroslava should use to achieve time t. Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
10 8.9 40075000 272.1	157079.04857106 15.598261092309

Sample Input 2	Sample Output 2
47 4.2 1337102.4 256	5836.2936298227 8.9113418228146

Problem F
Finding Lines
Time limit: 4 seconds

Annabel and Richard like to invent new games and play against each other. One day Annabel has a new game for Richard. In this game there is a game master and a player. The game master draws n points on a piece of paper. The task for the player is to find a straight line, such that at least p percent of the points lie exactly on that line. Richard and Annabel have very good tools for measurement and drawing. Therefore they can check whether a point lies exactly on a line or not. If the player can find such a line then the player wins. Otherwise the game master wins the game.

There is just one problem. The game master can draw the points in a way such that it is not possible at all to draw a suitable line. They need an independent mechanism to check whether there even exists a line containing at least p percent of the points, i.e., $\lceil n \cdot p/100 \rceil$ points. Now it is up to you to help them and write a program to solve this task.

Input

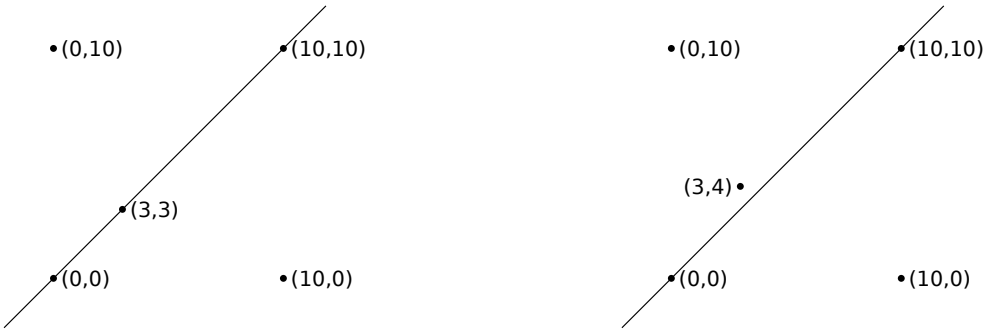
The input consists of:

- one line with one integer n ($1 \leq n \leq 10^5$), the number of points the game master has drawn;
- one line with one integer p ($20 \leq p \leq 100$), the percentage of points which need to lie on the line;
- n lines each with two integers x and y ($0 \leq x, y \leq 10^9$), the coordinates of a point.

No two points will coincide.

Output

Output one line containing either "`possible`" if it is possible to find a suitable line or "`impossible`" otherwise.



(a) Sample input 1: A line with (at least) 3 of the points exists.

(b) Sample input 2: No line with at least 3 points exists.

Figure F.1: Illustration of the sample inputs

Sample Input 1

```
5
55
0 0
10 10
10 0
0 10
3 3
```

Sample Output 1

```
possible
```

Sample Input 2

```
5
45
0 0
10 10
10 0
0 10
3 4
```

Sample Output 2

```
impossible
```

Problem G
Gathering
Time limit: 3 seconds

The citizens of Fictitia have had enough! The city keeps getting bigger and bigger, and all the more boring. Fictitia consists of horizontal and vertical streets only. The distance between each pair of neighboring parallel streets is always the same; we take this as the unit distance. Surely some variation could not hurt?

In order to draw more support and make their unhappiness known to the municipality, a group of citizens has agreed to gather at an intersection of the city to protest. The question is: which intersection? Since there is not much difference between them, the idea was raised to select an intersection (x^*, y^*) that minimizes the total distance everyone has to travel. Since everyone lives close to an intersection, the individual distance travelled by someone who lives at (x, y) is given by $|x - x^*| + |y - y^*|$.

Picture by Radosław Drożdżewski via Wikimedia Commons, cc by-sa

However, this could present a problem for the people who live far away, since they might have trouble getting there in time. Therefore it was decided that the intersection should be at most a certain distance d away from everyone. Given that restriction, can you help them identify an intersection that minimizes the total distance everyone has to travel?

Input

The input consists of:

- one line with one integer n ($2 \leq n \leq 100\,000$), the number of citizens;
- n lines each with two integers x and y ($0 \leq x, y \leq 10^9$), the coordinates of each citizen's house;
- one line with one integer d ($0 \leq d \leq 2 \cdot 10^9$), the maximum distance that each citizen should have to travel.

It is possible for multiple citizens to live at the same intersection.

Output

Output one line with a single integer: the smallest possible total distance that all citizens need to travel. If there is no intersection that everyone lives within a distance d of, output "`impossible`" instead.

Sample Input 1

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

Sample Output 1

```
18
```

Sample Input 2

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

Sample Output 2

```
20
```

Sample Input 3

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

Sample Output 3

```
impossible
```

Problem H
Hyacinth
Time limit: 3 seconds

As a new employee at the Northwestern Europe Routing Company (NWERC), you do a lot of thinking about wireless network architectures. Lately you learned about a multi-channel mesh network architecture (called *Hyacinth*) that equips each mesh network node with multiple network interface cards (NICs) to increase the network throughput. You can choose a channel frequency for each NIC. In order to communicate, for every two network nodes that are in range of each other, their NICs must share at least one common frequency. The theoretical throughput is optimal when the total number of used frequencies in the network is maximal.

Photo by Wikimedia Commons user The_wub

Your bosses at NWERC want you to figure out a procedure for assigning frequencies to the NICs such that the number of frequencies in use is maximized, subject to the constraint that all pairs of adjacent nodes must be able to communicate. A frequency is considered used if any pair of nodes within range of each other share that frequency. In the mesh network that you will be dealing with, each node is equipped with exactly two NICs (i.e., each node can use at most two frequencies). Since you are new at NWERC, your bosses further restrict the network layouts to make your job easier: the network graph will form a tree.

Input

The input consists of:

- one line with one integer n ($2 \leq n \leq 10\,000$), the number of nodes in the network;
- $n-1$ lines, each with two space-separated integers i and j, with $1 \leq i, j \leq n$ signifying that the (one-indexed) network nodes i and j are in range of each other.

Output

Output a frequency assignment for each of the $2n$ NICs such that all adjacent nodes can communicate and the number of used frequencies is maximized. You should output n lines, where the ith line contains the two frequencies of network node i. Valid frequencies are nonnegative integers less than 10^9.

Sample Input 1	Sample Output 1
2 1 2	23 42 42 23

Sample Input 2	Sample Output 2
14 1 2 1 3 1 4 2 5 2 6 3 7 4 8 4 9 4 10 7 11 7 12 7 13 7 14	4711 815 666 4711 4711 42 815 7 47 666 666 54 23 42 7 2 7 1 7 3 23 4 42 5 23 6 42 8

Problem I
Indoorienteering
Time limit: 9 seconds

Lukáš really likes orienteering, a sport that requires locating control points in rough terrain. To entertain the NWERC participants Lukáš wants to organize an orienteering race. However, it would be too harsh for the participants to be outdoors in this cold Swedish November weather, so he decided to jump on the new trend of indoor races, and set the race inside the B building of Linköping University.

Lukáš has already decided on the locations of the control points. He has also decided on the exact length of the race, so the only thing remaining is to decide in which order the control points should be visited such that the length of the total race is as he wishes. Because this is not always possible, he asks you to write a program to help him.

Note from the organizer: the NWERC indoorienteering race for this year has been cancelled since we neglected to apply for an orienteering permit in time from the university administration. (We still need you to solve the problem so that we can organize it for next year.)

Fredrik and Tommy lost in the B building. Photo by Tommy Olsson. cc-by-sa

Input

The input consists of:

- one line with two integers n ($2 \leq n \leq 14$) and L ($1 \leq L \leq 10^{15}$), the number of control points and the desired length of the race, respectively;
- n lines with n integers each. The jth integer on the ith line, d_{ij}, denotes the distance between control point i and j ($1 \leq d_{ij} \leq L$ for $i \neq j$, and $d_{ii} = 0$). For all $1 \leq i, j, k \leq N$ it is the case that $d_{ij} = d_{ji}$ and $d_{ij} \leq d_{ik} + d_{kj}$.

Output

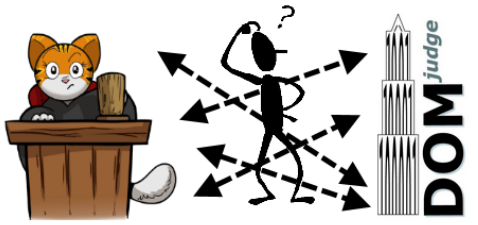
Output one line with "`possible`" if it is possible to visit all control points once in some order and directly return to the first one such that the total distance is exactly L, and "`impossible`" otherwise.

Sample Input 1	Sample Output 1
4 10 0 3 2 1 3 0 1 3 2 1 0 2 1 3 2 0	possible

Sample Input 2	Sample Output 2
3 5 0 1 2 1 0 3 2 3 0	impossible

Problem J
Judging Troubles
Time limit: 4 seconds

The NWERC organisers have decided that they want to improve the automatic grading of the submissions for the contest, so they now use two systems: DOMjudge and Kattis. Each submission is judged by both systems and the grading results are compared to make sure that the systems agree. However, something went wrong in setting up the connection between the systems, and now the jury only knows all results of both systems, but not which result belongs to which submission! You are therefore asked to help them figure out how many results could have been consistent.



Kattis and DOMjudge

Input

The input consists of:

- one line with one integer n ($1 \leq n \leq 10^5$), the number of submissions;
- n lines, each with a result of the judging by DOMjudge, in arbitrary order;
- n lines, each with a result of the judging by Kattis, in arbitrary order.

Each result is a string of length between 5 and 15 characters (inclusive) consisting of lowercase letters.

Output

Output one line with the maximum number of judging results that could have been the same for both systems.

Sample Input 1	Sample Output 1
5 correct wronganswer correct correct timelimit wronganswer correct timelimit correct timelimit	4

This page is intentionally left (almost) blank.

Problem K
Knapsack Collection
Time limit: 4 seconds

Gerald's job is to welcome the teams for this year's NWERC at the airport in Linköping. One of his duties is to stand at the luggage carousel and collect all the knapsacks that the teams are bringing. Gerald is a lazy person, so he just stands at the same position of the carousel and waits for bags to pass by so he can pick them up.

Picture by Bernhard J. Scheuvens via Wikimedia Commons.

The baggage carousel consists of s luggage slots, numbered in ascending order from 0 to $s-1$. Since the baggage carousel is cyclic, luggage slots $s-1$ and 0 also lie side by side. The carousel turns in such a way that if Gerald stands in front of slot i at some point in time, he will stand in front of slot $(i+1) \bmod s$ one time unit later.

In the beginning Gerald prepares a huge baggage cart at some position and stands there to wait for luggage. When a knapsack arrives in front of Gerald, he needs t time units to take it and put it on the baggage cart. After these t time units he is ready to pick up another knapsack. As long as there are remaining knapsacks on the luggage carousel, Gerald always takes the next one to arrive at his position as soon as he is ready after putting away the previous one.

Now Gerald wonders about the effect of his choice of position on the time it will take him to finish this task. It is up to you to help Gerald calculate the minimum, maximum, and average time to pick up all knapsacks, taken over all s possible slots, which can appear in front of Gerald after preparation. Time starts when he has prepared the baggage cart at some slot of the baggage carousel and ends after he has put the last knapsack on the cart.

Input

The input consists of:

- one line with three integers n ($1 \leq n \leq 2\,000$), s ($1 \leq s \leq 10^7$) and t ($1 \leq t \leq 10^7$), where n is the number of knapsacks to pick up, s is the number of slots of the carousel, and t is the number of time units Gerald needs to pick up a knapsack from the carousel and put it on the cart;
- one line with n integers $k_1, \ldots, k_n$ ($0 \leq k_i \leq s-1$ for $1 \leq i \leq n$), the slots of the knapsacks.

There may be several knapsacks stacked on top of each other in the same slot, but Gerald can still only pick up one knapsack at a time.

Output

Output three lines of output containing the minimum, maximum, and average time to pick up all the luggage, over all s positions. The average time should be output as a reduced fraction in the form p/q.

Sample Input 1	Sample Output 1
7 10 10000000 0 0 0 0 0 0 1	70000001 70000009 350000027/5

Sample Input 2	Sample Output 2
10 10 3 0 0 2 2 4 4 6 6 8 8	39 40 79/2

Sample Input 3	Sample Output 3
9 10000000 1 0 7 2 3 4 5 6 1 8	9 10000000 12500021249991/2500000