1)Fruit Bowl

Problem Description

Samaira is playing a mobile game where she needs to connect dots to form a "bowl" that catches falling "fruits".

The Challenge:

- **Bowl Creation:** Connect specific dots to create a bowl shape.
- **Fruit Catching:** The bowl must be designed so that designated "fruit" dots fall directly into it without hitting the edges and being deflected.

Your Task: Calculate the perimeter of the bowl formed by connecting the dots.

- **Accuracy:** Round the calculated perimeter to the nearest integer. For example, if the perimeter is 3.5, round it to 4.

Note: The distance between two coordinates is calculated by using formula:

sqrt ([$(x_2 - x_1)^2 + (y_2 - y_1)^2$]) where, ($x_1$, y_1) and (x_2, y_2) are the two coordinates.

Constraints

4 <= N <= 50

Input

First line contains 'N' defining number of dots scattered on the screen.

Next 'N' lines contain the coordinates (x, y space-separated integers) of scattered 'N' dots.

Output

Single line integer (rounded up to the nearest integer of the calculated length).

Time Limit (secs)

1

Examples

Example 1

Input

8

3 4

5 5

2 2

2 5

6 4

4 1

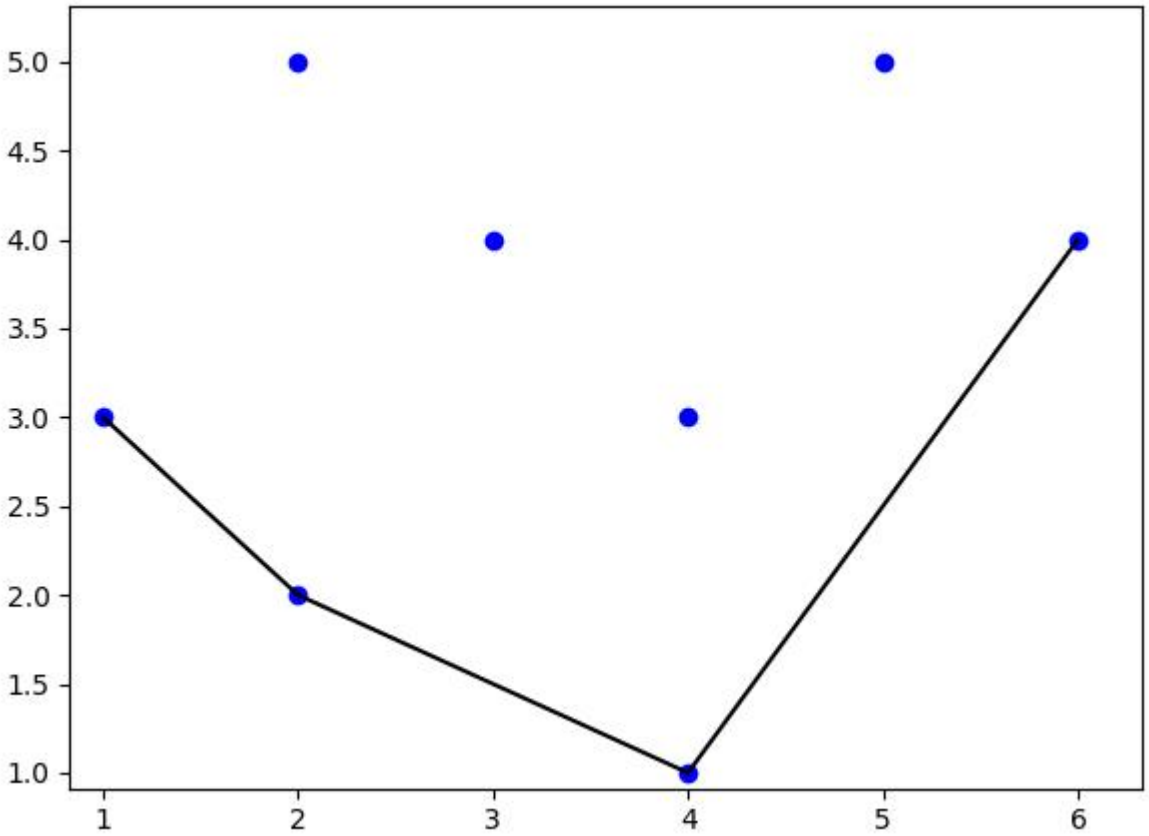
1 3

4 3

Output

7

Explanation



The above structure describes the bowl made in this level. The dots - (1, 3), (2, 2), (4, 1) and (6, 4) can be connected to build a bowl-like shape and every other dot will directly fall into the structure made. Since the connected dots can define a bowl, the calculated perimeter might be in fractional number. The perimeter calculated will be 7.26 and thus its nearest integer will be 7.

Example 2

Input

14

3 9

4 9

7 9

2 7

5 7

9 6

1 5

6 5

7 0

3 5

3 1

2 3

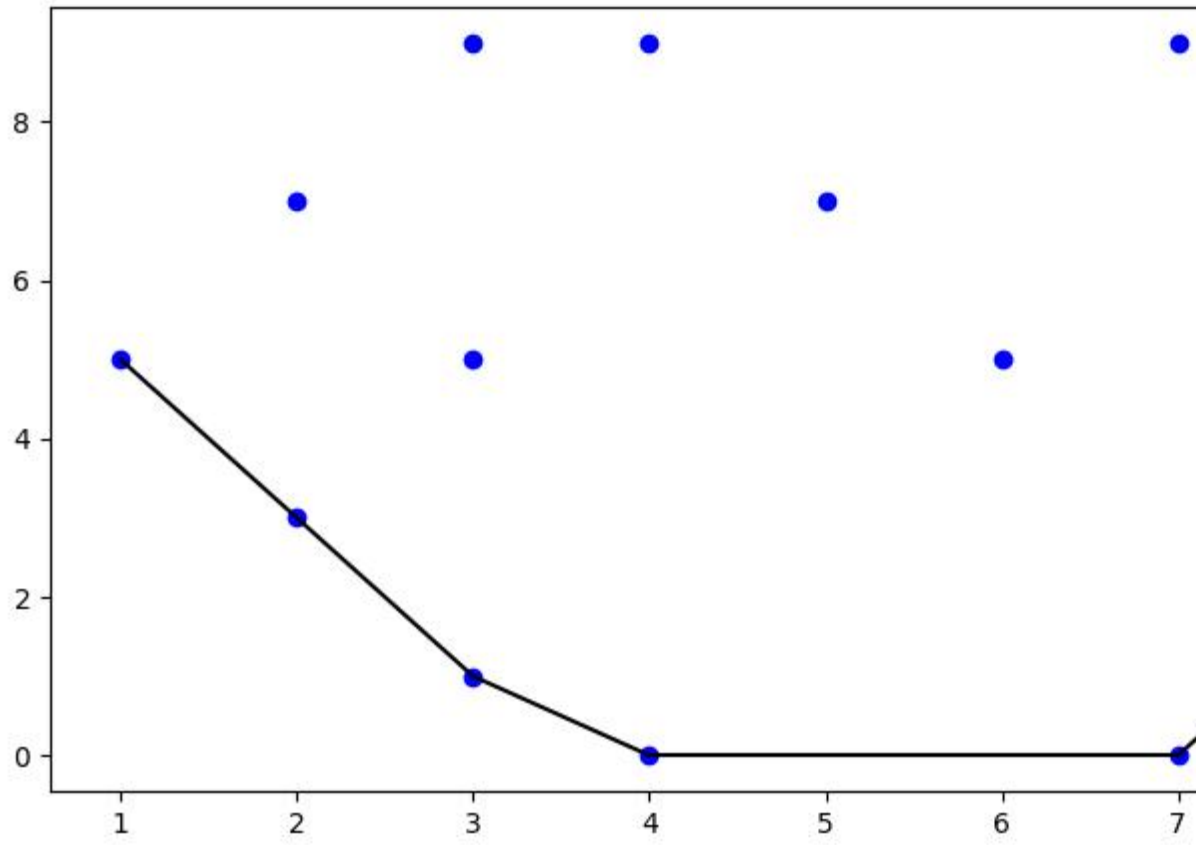
8 2

4 0

Output

15

Explanation



The above structure describes the bowl made in the level. The dots - (1, 5), (3, 1), (4, 0), (7, 0), (8, 2) and (9, 6) can be connected to create a bowl-like shape whereas rest of the dots will represent 'fruits' which will fall into the bowl. Therefore, the perimeter of the created bowl will be calculated as 15.25 and thus its nearest integer will be 15.

2)

Min Office Hours

Problem Description

An IT organization is trying to improve employee productivity by implementing mandatory work hours. However, employees are finding ways to avoid these hours, such as taking extended breaks or "tailgating" (following others through entry systems without swiping their ID cards). This leads to difficulties in accurately tracking work time.

The manager of a specific project wants to identify the employee who spends the least amount of time working to implement a penalty system. He suspects one employee but needs help verifying if this suspicion is accurate.

Data:

- **Log Records:** A daily log on a particular day of employee ID card swipes at various locations, including:

- Rooms (e.g., room1, room2...roomN)
- Cafeterias (e.g., cafeteria1, cafeteria2...cafeteriaM)
- Pantries (e.g., pantry1, pantry2...pantryN)

Challenge: Analyse the log data to determine:

- **Actual Work Time:** Calculate the time spent working by each employee in any room, excluding breaks and time spent in pantries and cafeterias.
- **Suspect Verification:** Confirm if the suspected employee truly spends the least amount of time working compared to all other project members.

Constraints

5 < N < 200

10 < Employee ID < 1000

Input

1. **Line 1: N** - This integer represents the total number of log entries recorded by the system.
2. **Lines 2 to N+1: Log Entries** - Each line describes a single employee activity and follows this structure:

- **EmployeeID**: An integer representing the unique ID of the employee.
- **Activity**: A string indicating either "enters" or "exits".
- **Location**: A string specifying where the activity occurred (e.g., "room1", "cafeteria2", "pantry3").
- **Time**: A string representing the time of the activity in 12-hour format (e.g., "10:30 AM").

1. **Line N+2: Suspect EmployeeID** - This line contains a single integer, the ID of the employee the manager is suspicious about.

Output

Single line consisting of 'Yes/No/Cannot be determined'.

Yes - if the manager's suspicion is correct

No - if the manager's suspicion is incorrect

Cannot be determined - if it is not possible to determine whether the suspected employee had worked for the least duration or not.

Time Limit (secs)

1

Examples

Example 1

Input

10

101 enters cafeteria1 9:01AM

101 exits cafeteria1 10:17AM

154 enters room5 9:15AM

153 exits room4 11:40AM

153 enters pantry2 2:00PM

101 enters room4 12:01PM

154 exits room5 1:05PM

153 exits pantry2 4:00PM

153 enters room4 10:16AM

101 exits room4 1:25PM

153

Output

Yes

Explanation

Analysis:

- Based on the 10 log entries provided, employees with ID 101 and 153 have the shortest working duration at 1 hour and 24 minutes. This calculation only considers time spent in workrooms (e.g., "room1", "room3") and excludes time in cafeterias or pantries.
- The manager is specifically questioning employee ID 153's actual work time.

Answer: Yes, the manager has a valid reason to doubt EmployeeID 153 as being the one with least working hours.

Example 2

Input

10

101 enters cafeteria1 9:01AM

101 exits cafeteria1 10:17AM

154 enters room5 9:15AM

153 exits room4 11:20AM

153 enters pantry2 2:00PM

101 enters room6 12:01PM

154 exits room5 1:05PM

153 exits pantry2 4:00PM

153 enters room4 10:16AM

101 exits room4 1:49PM

153

Output

Cannot be determined

Explanation

Analysis:

- The system recorded only 10 logs in total.
- Employee ID 101's work time is unreliable due to missing entry and exit logs.
- We only have records of them entering room 4 and exiting room 6, suggesting potential tailgating (following another employee through the entry system without swiping their own ID card).

Conclusion:

Because of incomplete log data for EmployeeID 101, we cannot definitively determine whether they worked for the shortest duration compared to other employees. The missing logs raise questions about their actual work time.

3) Cargo Shipping

Problem Description

At a busy airport, a conveyor belt system is used to unload goods from various flights parked at different gates and deliver them to a central warehouse. The conveyor belt network has multiple junctions between the gates and the warehouse. Each junction connects to multiple paths but can only keep one path open at a time. If a new path is opened at a junction, any previously open path is automatically closed.

The warehouse is also considered a junction; therefore, all locations with multiple incoming paths are referred to as junctions.

However, there is a limit to how many times a path can be switched at each junction, denoted by K. If a junction exceeds its switching limit, it gets temporarily locked, preventing any further changes to its paths. In this case, the goods from the affected flight must be manually carried to the warehouse, incurring a cost of *array[i]* rupees for the i^{th} item.

Example: If Flight A unloads its goods at Gate A, the conveyor is directed through the necessary junctions to connect Gate A to the warehouse. When Flight B at Gate B needs to unload its goods, the conveyor paths at the junctions are adjusted to redirect from Gate B to the warehouse, counting as switches. If the number of switches at any junction exceeds K, the system halts for that junction (i.e., whatever path is opened previously will remain open and cannot be further switched), and the goods must be manually transported.

Given:

- The conveyor belt network connecting the warehouse to all gates where flights are located.
- A sequence indicating which flight each incoming good is coming from.
- An array representing the cost of manually shifting the i^{th} good to the warehouse when they are stuck.

Calculate and print the total manual shifting cost.

Constraints

1 <= number of goods <= 50

1 <= number of junctions <= 50

1 <= length of name of flights and junctions<= 20

1 <= K <= 10

1 <= Cost of manual shipping of each item to warehouse <= 100

There will always be a single warehouse.

There will be only one gate for each flight.

Name of the junctions and flights will consist of lower-case alphabets and digits.

Process the given goods sequentially from left to right.

Irrespective of at which junction the good is struck, the manual cost of traversing the good to the warehouse remains same i.e., as given in the input.

Input

- The first line contains an integer *N,* representing the number of lines describing the entire conveyor belt system. These lines will be provided in a random order.
- Each of the following *N* lines represents a segment of the conveyor structure. Each line follows this format:
    - The first element is a string representing a junction ID.
    - Subsequent elements are the strings representing connections to other junctions or gates. Remember that paths flow in one direction from gates to the warehouse.
- The next line lists the gate names from which each item originates, separated by spaces.
- The second-to-last line contains an array of integers representing the cost of manually shifting each item to the warehouse when a junction is locked. These costs are separated by spaces.
- The final line contains an integer *K*, indicating the maximum number of switches allowed at any junction (inclusive).

Output

Output the total manual shifting cost required to move all items from all gates to the warehouse.

Time Limit (secs)

1

Examples

Example 1

Input

6

warehouse jun1 jun3

jun1 jun2

jun2 jun4 jun6

jun4 flyhigh

jun6 jetstream falconflies

jun3 bharataero

flyhigh falconflies bharataero jetstream flyhigh bharataero

4 2 9 15 7 3

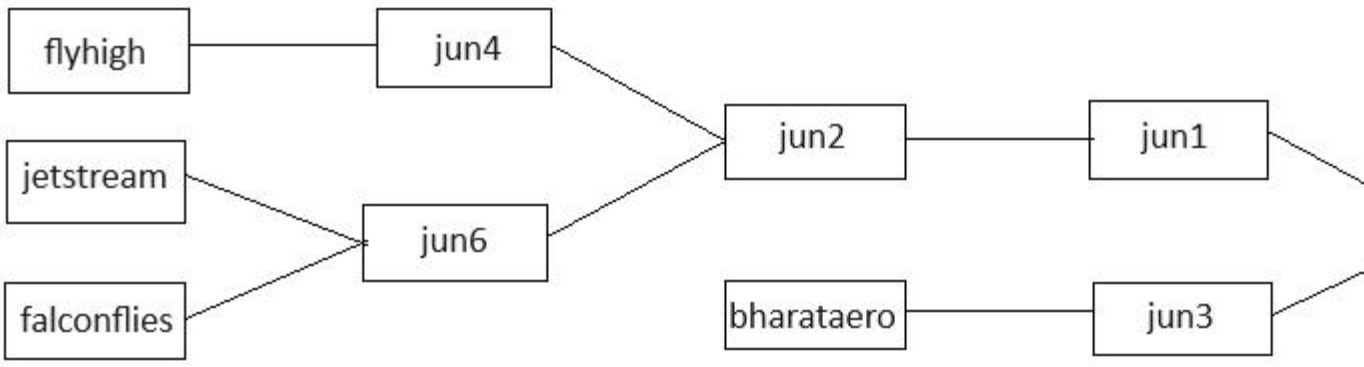
2

Output

22

Explanation

The given structure looks like below.



Initially, all paths are closed. Here is an overview of the paths connected to each junction:

- jun4: Path from flyhigh
- jun6: Paths from jetstream and falconflies
- jun2: Paths from jun4 and jun6
- jun1: Path from jun2
- jun3: Path from bharataero
- warehouse: Paths from jun1 and jun3

Sequence of Operations:

1. The first item arrives from flyhigh. To ship it to the warehouse, the following path is opened:
   flyhigh -> jun4 -> jun2 -> jun1 -> warehouse.
   Each junction of this path is switched (or opened) once.

2. The next item is from falconflies. The required path is:
   falconflies -> jun6 -> jun2 -> jun1 -> warehouse.
   Each junction is switched. So far, the paths switched are:

- jun4: 1 time
- jun6: 1 time
- jun2: 2 times
- jun1: 1 time
- warehouse: 1 time

3. The third item is from bharataero. The path is:
   bharataero -> jun3 -> warehouse.
   Only the paths at jun3 and warehouse needs to be switched.

- jun4: 1 time
- jun6: 1 time
- jun2: 2 times
- jun1: 1 time
- jun3: 1 time
- warehouse: 2 times

4. The fourth item is from jetstream. The required path is:
   jetstream -> jun6 -> jun2 -> jun1 -> warehouse.
   The path at warehouse needs to be switched but it has already reached its maximum switching limit. Hence, we need to carry this item manually, which will incur a cost of 15.

5. The fifth item arrives from flyhigh. The path is:
   flyhigh -> jun4 -> jun2 -> jun1 -> warehouse.
   But the warehouse has reached its maximum switching limit. This item must also be manually transported, adding a cost of 7.

6. The final item is from bharataero. The path is: bharataero -> jun3 -> warehouse.
   Since all required paths are already open, no manual shifting is needed.

Total Manual Shifting Cost is: 15 + 7 = 22. Hence print the same.

Example 2

Input

4

warehouse jun1

jun1 thunderbird jun2 jun3

jun2 aeronova

jun3 jetstream starwing

jetstream starwing aeronova jetstream jetstream thunderbird aeronova jetstream thunderbird aeronova

9 6 3 12 15 7 2 6 3 4

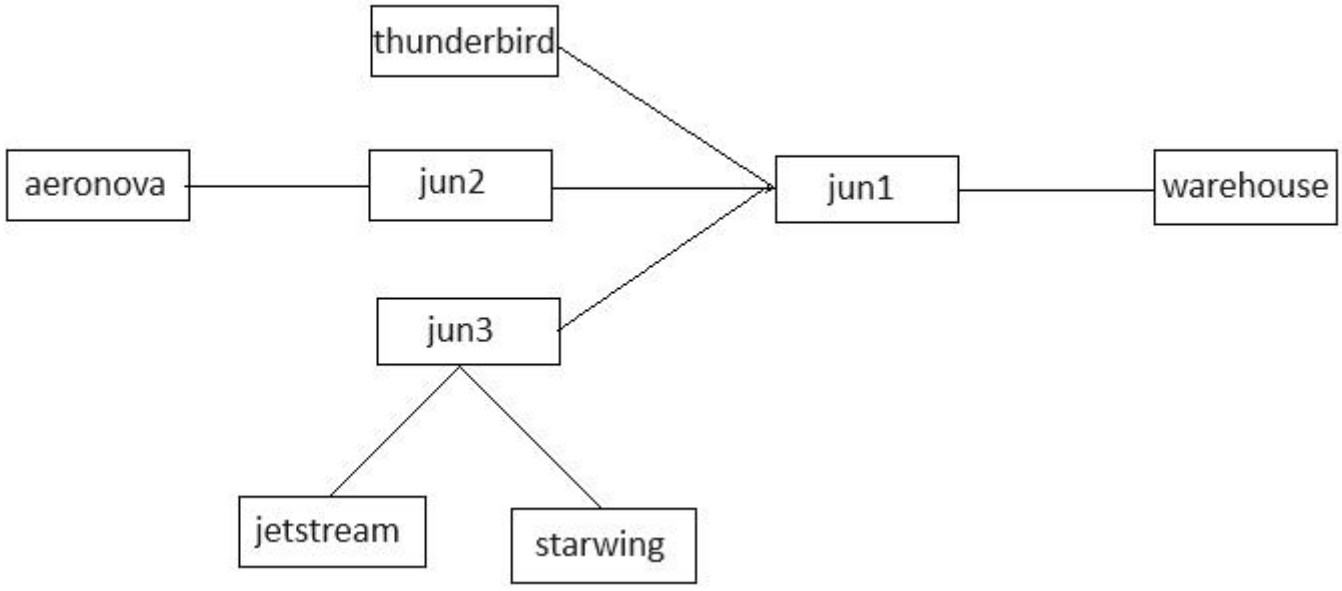
3

Output

16

Explanation

The given structure looks like below.



Initially, all paths are closed. Here is an overview of the paths connected to each junction:

- warehouse: path from jun1

- jun1: paths from thunderbird, jun2, and jun3
- jun2: path from aeronova
- jun3: paths from jetstream and starwing

Sequence of Operations:

1. jetstream: To ship the item from jetstream, the path jetstream -> jun3 -> jun1 -> warehouse is opened. Each junction is switched.
2. starwing: The item from starwing follows the same path: starwing -> jun3 -> jun1 -> warehouse. All required junctions are switched.

- jun3: 2 times
- jun1: 1 time
- warehouse: 1 time

3. aeronova: The item from aeronova takes the path aeronova -> jun2 -> jun1 -> warehouse. Each junction is switched.

- jun3: 2 times
- jun1: 2 times
- warehouse: 1 time
- jun2: 1 time

4. jetstream: The next item from jetstream again follows the path jetstream -> jun3 -> jun1 -> warehouse. The junctions are switched as needed.

- jun3: 3 times and
- jun1: 3 times

5. jetstream: Another item from jetstream uses the same path. However, the path is already open and nothing else is needed to be switched.
6. thunderbird: The item from thunderbird takes the path thunderbird -> jun1 -> warehouse. But the switching limit at jun1 has already reached and cannot be switched. Hence, this item needs to be carried manually which will incur a cost of 7.
7. aeronova: Another item from aeronova follows the path aeronova -> jun2 -> jun1 -> warehouse. However, the switching limit for jun1 is reached, so the item is manually transported, incurring a cost of 2.
8. jetstream: The next item from jetstream again uses the path jetstream -> jun3 -> jun1 -> warehouse. However, the path is already open and nothing else is needed to be switched.
9. thunderbird: For this item to reach the warehouse, the jun1 need to be switched which is impossible, hence manual shifting is required incurring a cost of 3.
10. aeronova: For this item to reach the warehouse, the jun1 need to be switched which is impossible, hence manual shifting is required incurring a cost of 4.

Total Manual Shifting Cost is: 7 + 2 + 3 + 4 = 16. Hence print the same.

4)

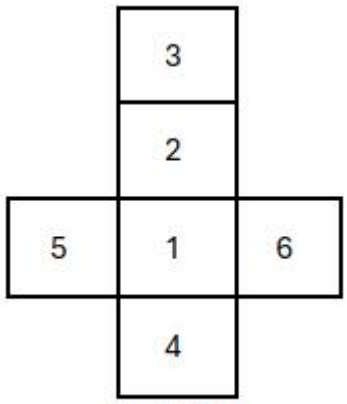
Color Box

Problem Description

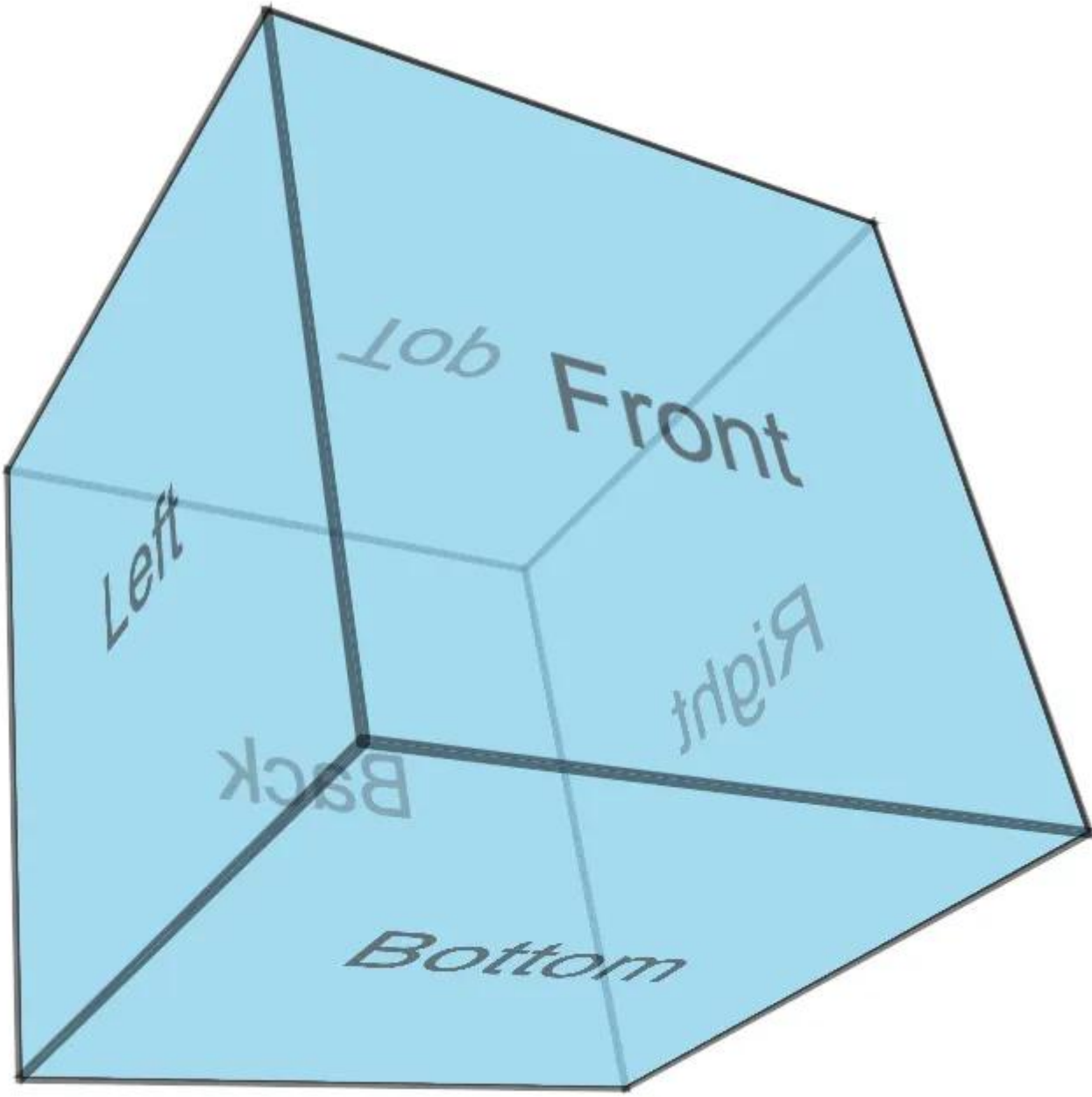
Ziva's Halloween Break: A Cube Colouring Conundrum.

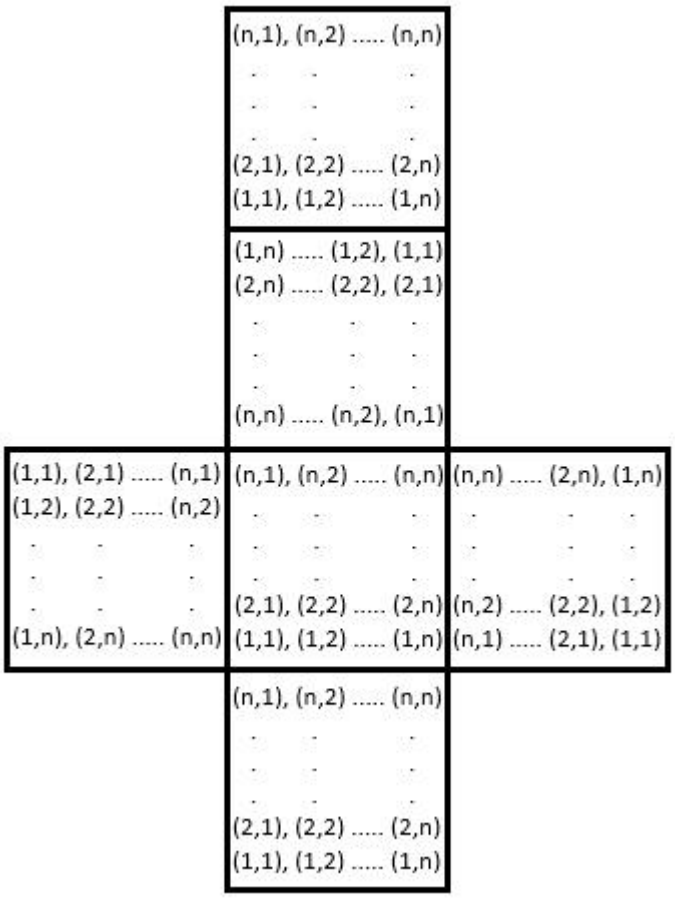
Ziva is bored during her Halloween break and decides to paint some pictures from her drawing book. To make things more interesting, she picks up a cube with sides of length N x N (imagine squares on each side!).

Feeling spontaneous, Ziva throws the cube into the air and catches it, letting it rotate randomly. Now, looking at one of the cube's sides at eye level, she wants to colour just one position on that side.



Open Cube





Faces of the cubes are described according to the numbers.

1 - base, 2 - back, 3 - top, 4 - front, 5 - left, 6 - right.

Rules:

1. 'turn left' - front goes to left, left goes to back, back goes to right, right goes to front, top side will be rotated right, whereas base side will be rotated left.

2. 'turn right' - front goes to right, right goes to back, back goes to left, left goes to front, top side will be rotated left, whereas base side will be rotated right.

3. 'rotate front' - front goes to base, base goes to back, back goes to top, top goes to front, left side will be rotated right, whereas right side will be rotated left.

4. 'rotate back' - front goes to top, top goes to back, back goes to base, base goes to front, left side will be rotated left, whereas right side will be rotated right.

5. 'rotate left' - top goes to left, left goes to base, base goes to right, right goes to top, front side will be rotated left, whereas back side will be rotated right.

6. 'rotate right' - top goes to right, right goes to base, base goes to left, left goes to top, front side will be rotated right, whereas back side will be rotated left.

You are given the initial faces of the cube (after placing it in front of her eye level), the types of rotations the cube will take after throwing it in the air, and side, row, and column of the cube from which the colour

is chosen. The faces may have multiple colours defined by various alphabets. You must determine which colour will be chosen after the rotations took place.

Constraints

2 <= N <= 35

1 <= K <= 50

Colours on each side of the box consists of upper-case English alphabets only.

Input

First line consists of 2 integers - N & K space separated (where 'N' is the number of rows and columns on each face of the cube and 'K' is the number of rotations or turns it took before choosing colour)

Next 6*N lines, each N lines describes each face of the cube in the order described above with each line consisting of N space separated characters.

Next 'K' lines describe the types of rotations the cube will perform.

Last line describes the side, 1-indexed row and column which is chosen by Ziva.

Output

One character which describes the colour chosen by Ziva

Time Limit (secs)

1

Examples

Example 1

Input

2 1

B R

R G

W Y

W G

B Y

O O

Y R

O B

B R

Y W

O R

B W

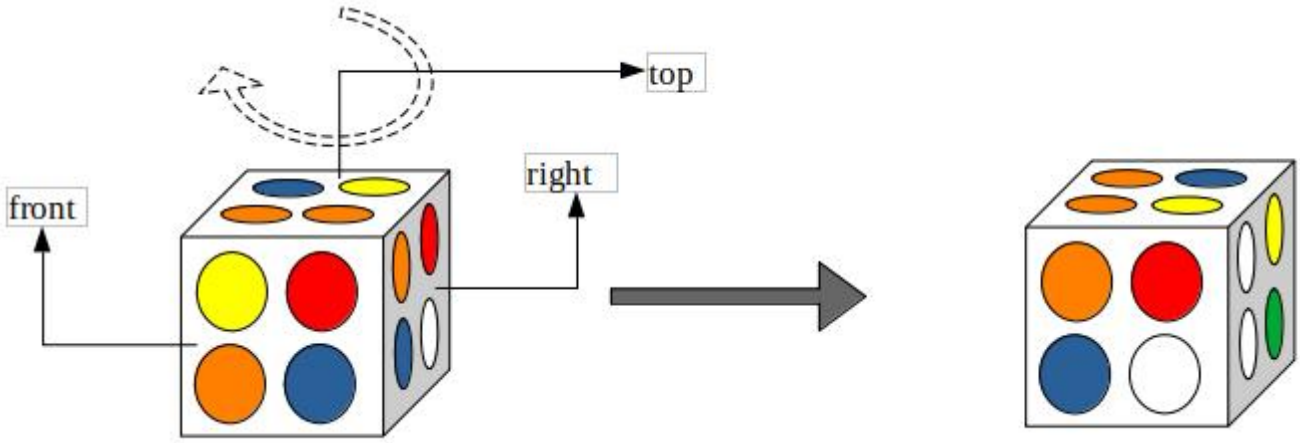
turn left

top 1 1

Output

O

Explanation



The box is rotated left. After the turn, the right side comes to front, and the colours of top face will change their position.

Example 2

Input

2 1

B R

R G

W Y

W G

B Y

O O

Y R

O B

B R

Y W

O R

B W

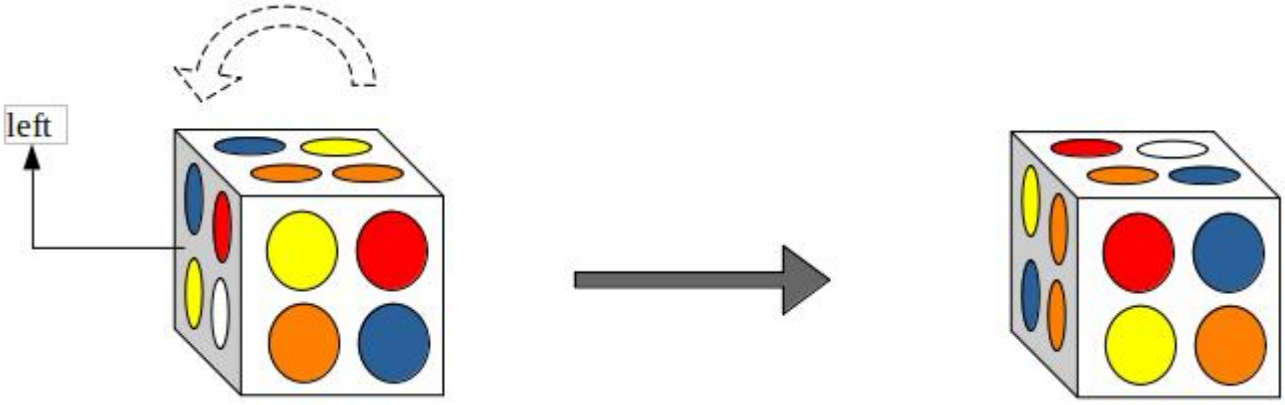
rotate left

left 1 1

Output

Y

Explanation



The box is turned left. After the rotation, the right side comes to top, and top will go in left. Now the colours of left side will change their position.

5)

Bands 2

Problem Description

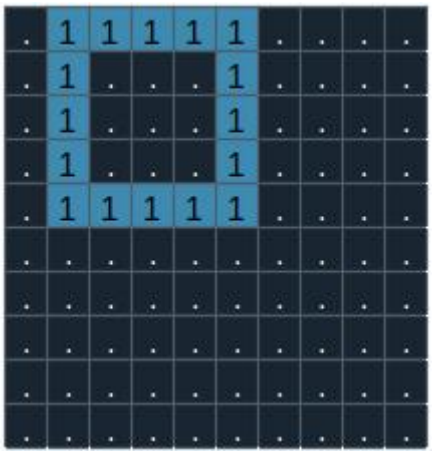
Two friends are shopping for friendship bands at a mall. They discover that each pair of bands comes in a transparent case where the bands are tangled together, impossible to separate without cutting!

To avoid ruining their bands, they decide to choose a pair that is properly intact. Each case is represented as a square matrix (size S x S). Each band within the case:

- Has a starting position within the matrix (coordinates).
- Is defined by a sequence of characters ("u," "d," "l," and "r") representing movements up, down, left, and right.
- Loops back to its starting position after following the entire sequence.

The top-left corner of the matrix is always index (0, 0).

For example, Assume the case size as 10, starting position is 0 1 and sequence rrrrddddllllluuuu. The diagram below illustrates this:



When placing the two bands in the case, assume each band occupies the cells sequentially from its starting position. The band placed first will lie beneath, while the second band will overlap it.

Given the placement details of two bands, help the friends determine whether the bands in the case can be separated without cutting either of them.

Constraints

5 < S < 25

Each band will overlap the other only, without overlapping itself.

Input

First line consists of a single integer S, representing the side of the square matrix.

Second line consists of two space separated integers denoting starting position of band 1.

Third line contains the sequence of characters "u," "d," "l," and "r" denoting the placement path of band 1.

Fourth line consist of two space separated integers denoting starting position of band 2.

Fifth line contains the sequence of characters "u," "d," "l," and "r" denoting the placement path of band 1.

Output

A single integer representing the maximum number of cells where one band overlaps the other if they can be separated without cutting;

Otherwise, print "Impossible."

Time Limit (secs)

1

Examples

Example 1

Input

10

0 1

rrrrddddlllluuuu

2 3

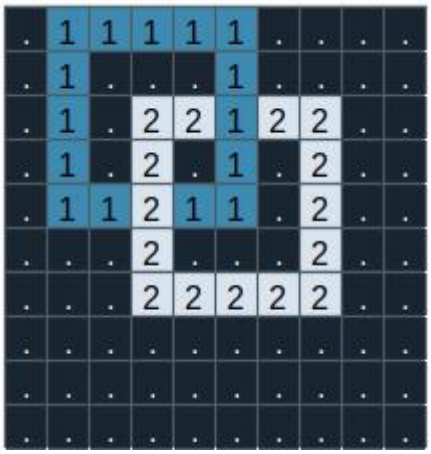
rrrrddddlllluuuu

Output

Impossible

Explanation

The image below depicts the input described above.



At the cell (2, 5) band2 will be placed first, later band1 will be placed above it.

At the cell (4, 3) band1 will be placed first, later band2 will be placed above it.

From the image you can see that the two bands cannot be separated without cutting any one of it. Hence print Impossible.

Example 2

Input

7

0 4

rddddllluuuurr

2 0

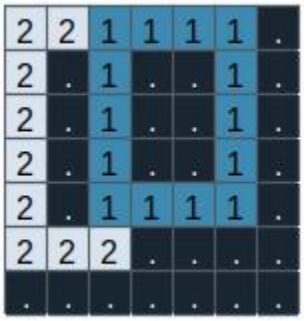
dddrruuuuulldd

Output

5

Explanation

The image below depicts the input described above.



Band 1 is placed entirely above Band 2, meaning they are not interlocked and can be separated without cutting. Band 2 overlaps Band 1 in 5 cells, while Band 1 does not overlap Band 2 at all. Therefore, the maximum overlap is 5, and the output is 5.

6)

Chess Key

Problem Description

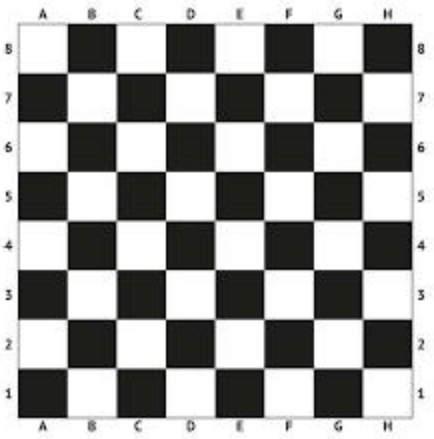
You're tasked with calculating the number of unique chessboard positions achievable after 'd' plies (one-half moves).

Here's the setup:

- **Chessboard:** Standard 8x8 grid.
- **Pieces:** At most three pieces of the same color: Queen, Rook, and Bishop.
- **Placement:** At least one of each piece must be present on the board.
- **Depth (d):** The number of plies (moves) to consider. A ply usually represents a half move in a 2-player game. In our case, since only one coloured pieces are placed, it should be construed to mean a move.

Your Goal: Determine the total number of unique chessboard positions possible after 'd' plies.

For better understanding the Chessboard, go through the details below.



The positions of the chess pieces are represented using three characters:

- The first character is either Q, B, or R, denoting a Queen, Bishop, or Rook, respectively.

- The second character ranges from A to H, indicating the column in the chessboard.
- The third character ranges from 1 to 8, indicating the row in the chessboard.

For example, QA8 represents a Queen positioned on the 8th row of the A column.

Constraints

$0 < d < 5$

Input

The first line specifies the positions of the pieces, separated by spaces.
The second line indicates d, the ply depth

Output

A single integer representing number of unique board positions across a depth of d plies

Time Limit (secs)

1

Examples

Example 1

Input

QA3 RB3

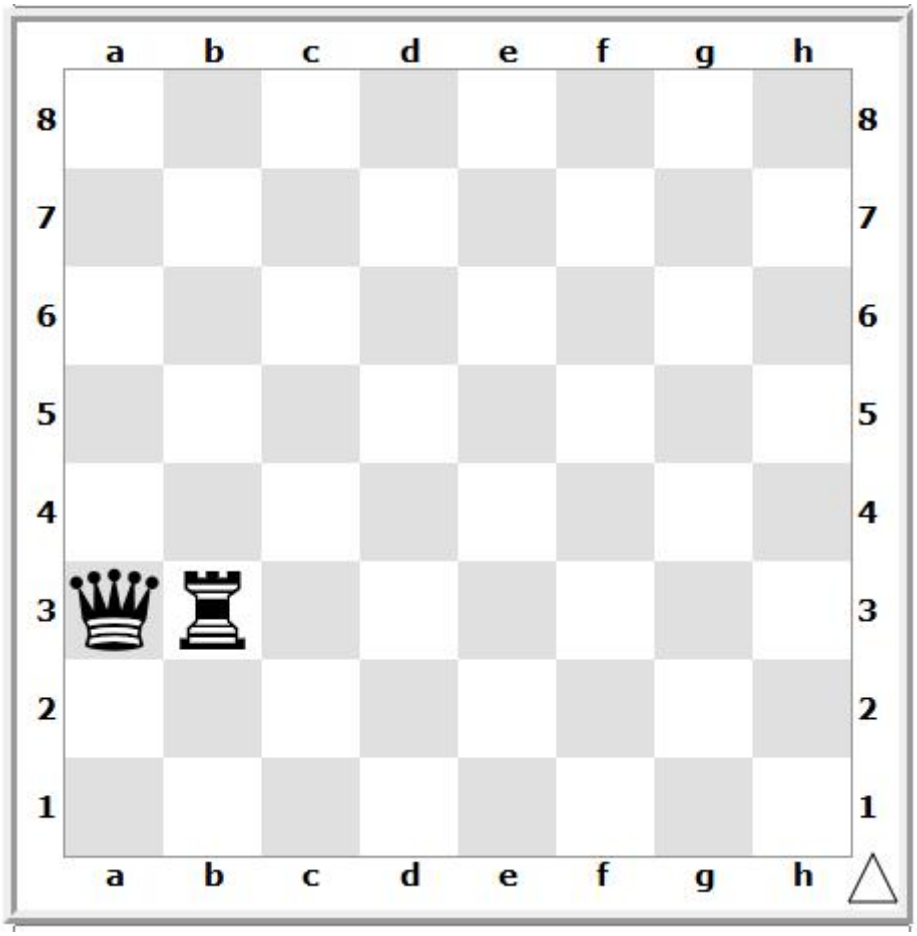
1

Output

27

Explanation

Below shows the initial state. After every move, the board position will be recorded.



In this chess position, either the Queen or the Rook will be moved on the next turn (one ply).

- **Queen's Movement:** The Queen can be legally placed in 14 distinct squares: a1, a2, a4 to a8, b2, c1, b4 to f8.
- **Rook's Movement:** The Rook can occupy 13 unique squares: b1, b2, b4 to b8, and c3 to h3.

Therefore, there are a total of 14 + 13 = 27 possible board positions after one ply (one move).

Print 27 as the output, representing the number of unique board positions achievable within a depth of 1 ply.

Example 2

Input

QA3

2

Output

64

Explanation

At a depth of 2 positions on an empty board (Queen is the only piece on the board), it will cover all 64 distinct squares. Hence only 64 unique positions are possible. Hence print 64 as output.

Example 3

Input

QA3 RB3

2

Output

388

Explanation

At a depth of 2 position with Queen at a3, Rook at b3, 388 is the number of unique board positions achievable within a depth of 2 ply.

7) Aarav And Arjun

Problem Description

Twin brothers Aarav and Arjun, both in eighth grade, love playing games that involve math and geometry. They already know about shapes, area, perimeter, and other geometric concepts.

Their mother wanted to create a fun challenge for them to practice their skills. She took some thin straight sticks and placed them on a 2D coordinate system, trying to form a closed shape. Aarav chose to work with the completed figure, while Arjun opted to use the leftover pieces of stick that did not form part of the closed shape.

Their mother then presented them with a series of tasks:

1. **Check for a Closed Figure:** First, both boys had to determine if a closed figure was formed by the arrangement made by their mother.
    a. **If a Closed Figure Exists:**
        i. Calculate its area.
        ii. See if Arjun could create the same shape and sized figure using the leftover sticks. The leftover sticks are the sticks that are not a part of the closed figure.

Note that Arjun can cut off any portion of a stick that is not part of the closed figure and use it to create the closed figure of same shape and size.

Whenever they completed a set of tasks, they had to follow this specific format as given in Output section when presenting their answers:

Constraints

1 <= N <= 20

0 <= x, y coordinates <= 50

There will not be more than one closed figure.

Extra lines will not intersect with lines of closed figure, but themselves they intersect.

Round off the results of distance of two points, intersection point to two decimal places.

Not more than two lines intersect at the same point.

Input

First line consists of *N,* denoting the number of sticks placed in the 2D coordinate system.

Next *N* lines consist of four space separated integers *x1 y1 x2 y2,* where (x1, y1) and (x2, y2) denotes the starting and ending points of the stick.

Output

Line 1: Print "Yes" or "No" indicating whether a closed figure is formed or not.

Line 2 (if Line 1 is "Yes"): Print "Yes" if Arjun can arrange the leftover stick pieces to form a figure that is both the same shape and the same size. Otherwise, print "No".

Line 3 (if Line 1 is "Yes"): Print the area of the closed figure rounded to two decimal places.

Time Limit (secs)

1

Examples

Example 1

Input

4

2 1 2 6

5 1 5 6

0 2 6 2

0 5 6 5

Output

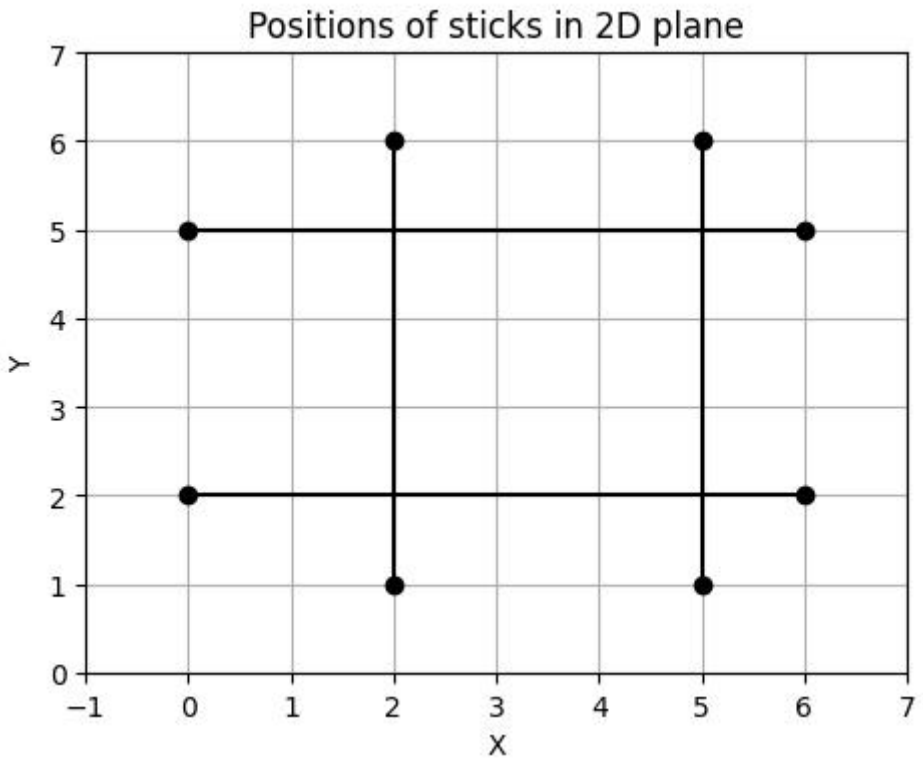
Yes

No

9.00

Explanation

The above input when plotted looks like below.



Here, you can see that a closed figure is formed. So, print "Yes" in the first line.

Next, as you can see that with the leftover sticks, Arjun cannot form the closed figure of same shape and size. Hence, print "No" on the second line.

And the area of the closed figure is 9, which we need to print up to 2 decimal points, hence print 9.00

Hence, the output look likes below -

Yes

No

9.00

Example 2

Input

5

4 1 4 5

5 0 1 4

1 2 5 6

0 0 3 1

5 2 5 5

Output

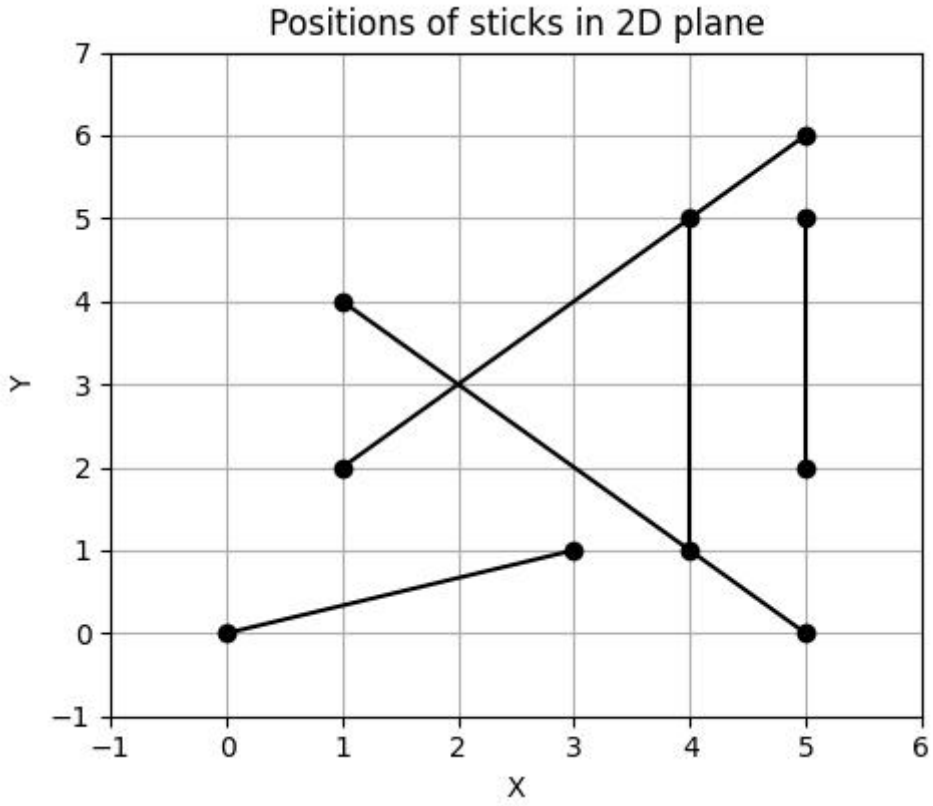
Yes

Yes

4.00

Explanation

The above input when plotted looks like below.



Here, you can see that a closed figure is formed. So, print "Yes" in the first line.

Next, as you can see that with the leftover sticks, Arjun can form the closed figure of same shape and size. Hence, print "Yes" on the second line.

And the area of the closed figure is 4, which we need to print up to 2 decimal points, hence print 4.00

Hence, the output look likes below -

Yes

Yes

4.00

Example 3

Input

6

0 4 0 6

1 2 4 4

5 5 5 1

1 6 4 4

1 4 3 1

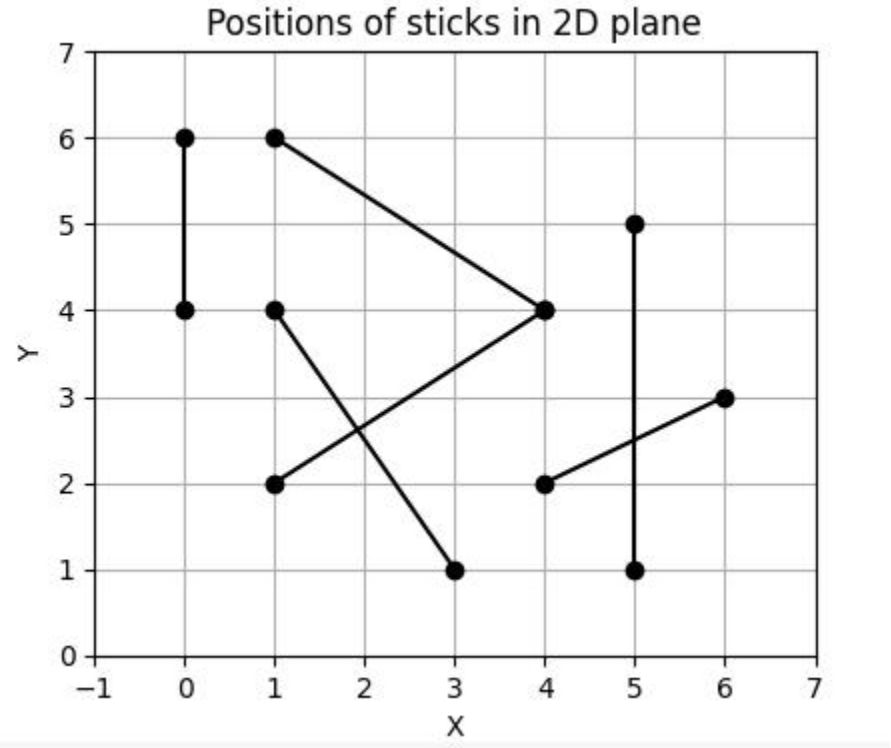
4 2 6 3

Output

No

Explanation

The above input when plotted looks like below.



Since there is no closed figure formed, print "No" and the second, third lines of output do not exist.

8) Bubble Trouble

Problem Description

Welcome to Bubble Land, an underwater world where everything is bubbly! You are navigating with a bubble-shaped vehicle through this fantastical place, starting at one point and aiming for another.

Here is the catch:

- **Bubble Buildings:** Structures are represented by their centre coordinates and radius, all enclosed within square borders.
- **Tax Lines:** Imaginary lines connect building centres, acting like toll roads. Crossing a tax line incurs a fee (touching does not!). These lines don't intersect.
- **Your Vehicle:** It is also bubble-shaped, with its own centre and radius.

Your Mission:

Find the path between your starting point and destination while avoiding:

- Overlapping any building.
- Crossing tax lines unnecessarily.

Navigate Bubble Land and minimize those annoying taxes!

Constraints

$5 < S <= 50$

$3 < N < 20$

$0 <= x, y <= S$

$0 <$ Radius of Buildings < 10

$1 <$ Radius of vehicle < 10

$0 <$ Tax lines < 20

Input

Line 1: A single integer **S**, representing the length of one side of Bubble Land (the size of the square area).

Line 2: Three space-separated integers:

1. **x**: The x-coordinate of your vehicle's centre.
2. **y**: The y-coordinate of your vehicle's centre.
3. **r**: The radius of your bubble vehicle.

Line 3: Two space-separated integers:

1. **x**: The x-coordinate of the destination.
2. **y**: The y-coordinate of the destination.

Line 4: A single integer **N**, representing the number of buildings in Bubble Land.

Lines 5 to N+4: Each line contains three space-separated integers:

1. **x**: The x-coordinate of a building's centre.
2. **y**: The y-coordinate of a building's centre.

3. **r**: The radius of the building.

Line N+5: A single integer **T**, representing the number of tax lines in Bubble Land.

Lines N+6 to N+T+5: Each line contains two space-separated integers:

1. : The index of bubble building of one end of a tax line (building centre).
2. : The index of bubble building the other end of a tax line .

Index of the building starts from 1 , index the buildings in the given order in input

Output

Single integer representing minimum number of tax lines to be crossed. If the destination is unreachable, print Impossible.

Time Limit (secs)

1

Examples

Example 1

Input

25

2 2 2

15 11

4

7 11 5

14 6 2

22 4 2

20 12 2

4

1 2

2 3

2 4

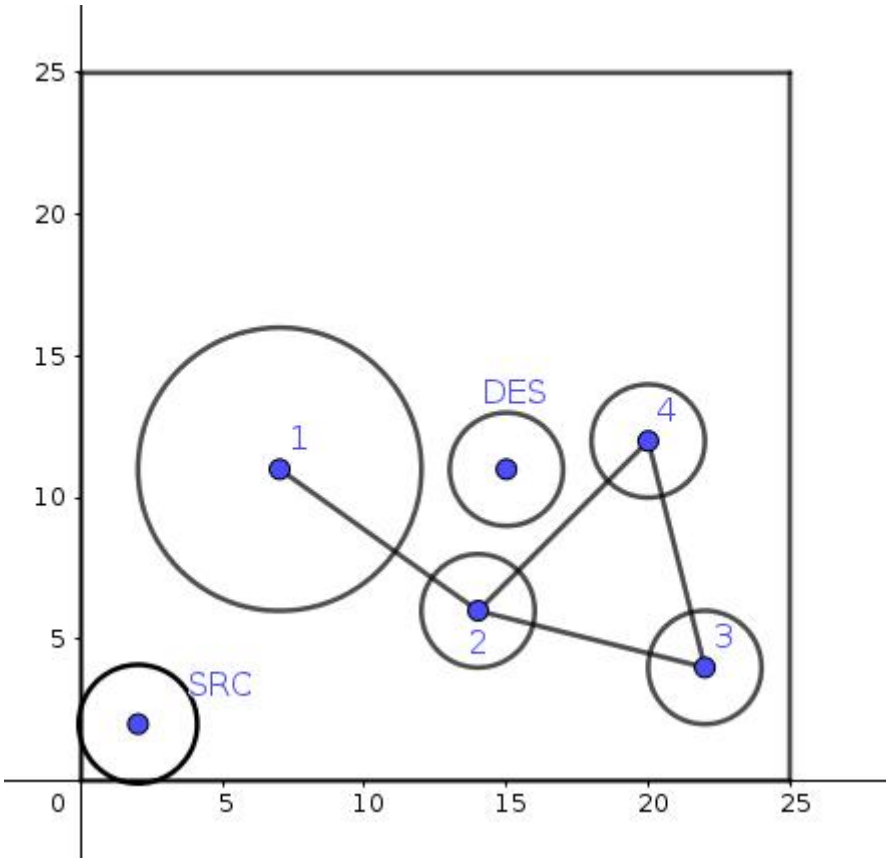
3 4

Output

2

Explanation

The below diagram represents the given input.



The path to reach destination without touching bubble building would be passing through tax lines drawn between buildings [2,3] and [2,4]. Here we are crossing 2 tax lines and that is the minimum possible. Hence, print the same.

Example 2

Input

25

4 4 3

21 21

4

5 17 4

10 9 4

15 16 4

20 9 4

3

1 2

2 3

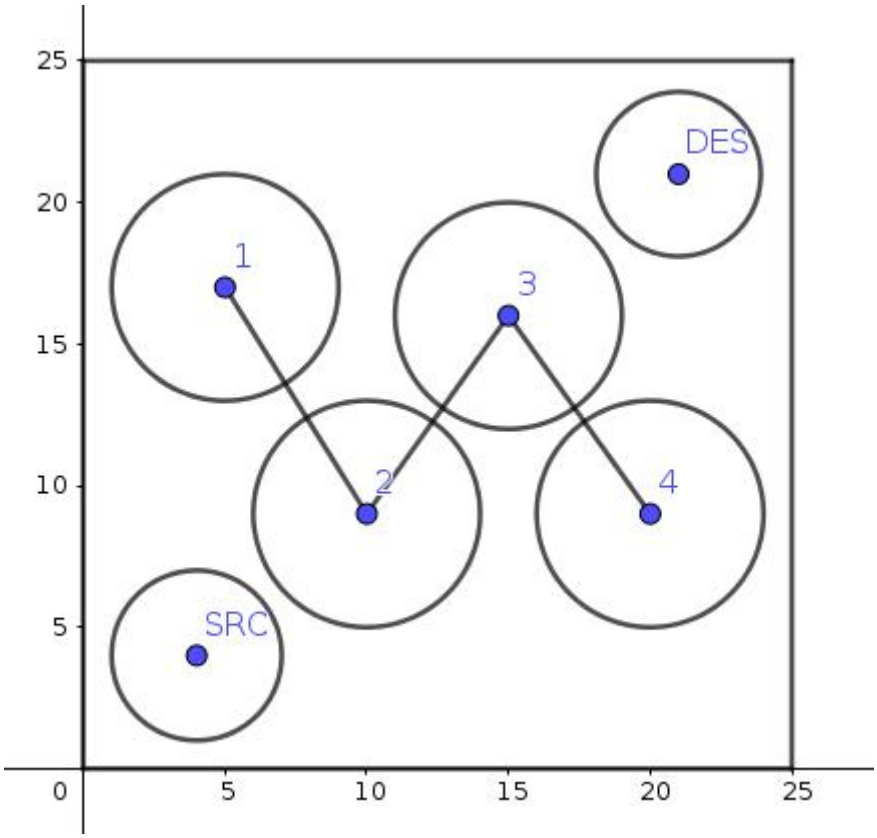
3 4

Output

Impossible

Explanation

The below diagram represents the given input.



From the image, it is not possible to reach destination without touching the bubble buildings. Hence print Impossible.