TCS CODEVITA

SEASON-13

The Brick wall

Problem Description

A plumber needs to install pipelines on a brick wall. To do this, some bricks must be broken to fit the pipes.

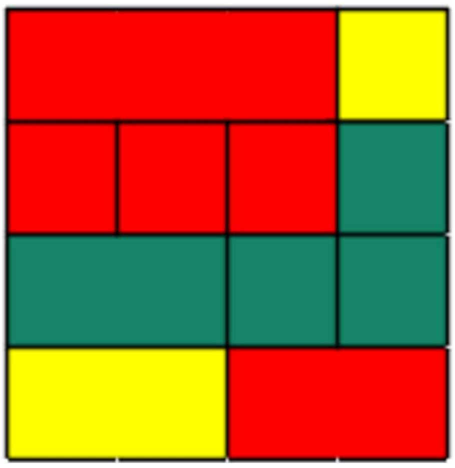
There are two types of bricks in the wall:

- **Red Bricks (R):** Hard to break.
- **Green Bricks (G):** Easy to break.

The plumber will only break Green Bricks to make the job easier. The wall is represented as a square grid, with each brick type and its length specified (e.g., "3R" means a Red Brick of length equal to three unit Bricks). The wall also includes a **Source (S)** where the pipe starts and a **Destination (D)** where it ends.

Pipes can be laid either vertically or horizontally, moving from the current brick to any adjacent Green Brick (up, down, left, or right). The goal is to find the minimum number of Green Bricks that must be broken to connect the source to the destination. Red Bricks cannot be used.

Assume the layout of the brick wall is shown below.



The input notation to represent the above brick wall is shown below:

```
3R1D
1R1R1R1G
2G1G1G
2S2R
```

Constraints

3<=N<=25

Input

The first line contains N, the size of the wall (N x N).
The next N lines describe the wall layout using the notation above.

Output

Print a single integer: the least number of Green Bricks that need to be broken.

Time Limit (secs)

1

Examples

Example 1

4

3R1D

1R1R1R1G

2G1G1G

2S2R

Output

4

Explanation

The input and image already shown in description.

From the image we can see that 4 bricks need to be destroyed to lay pipes between source and destination.

Example 2

Input

5

3G1R1G

1G1R1G2R

1S1R1G1R1D

2R1G1R1G

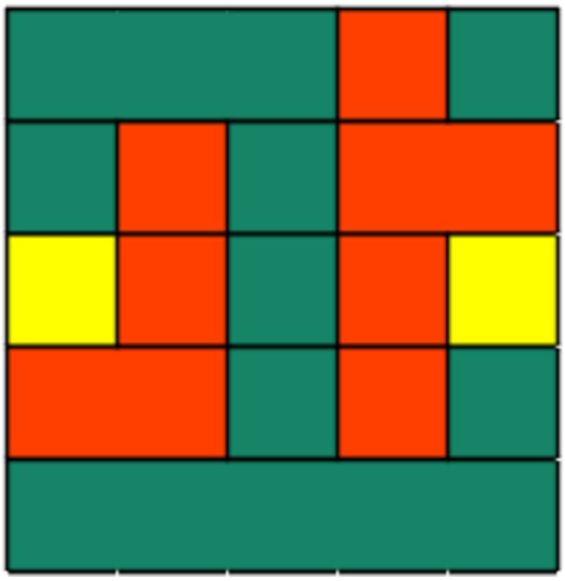
5G

Output

7

Explanation

The image below represents the above input.



From this you can see that 7 bricks must be broken to lay pipelines between source and destination.

TwoScouts

Problem Description

Two elite scouts, Arin and Bran, are stationed in different towns. Their mission: each must deliver an encrypted message to the main outpost.

In the distant land of Graphonia, towns are connected by a network of roads. Communication between these towns is crucial for survival. The main outpost has become the hub for vital resources and intelligence.

Due to the sensitive nature of their information, the scouts must take separate routes to the outpost. They can only meet at the outpost—nowhere else. Each scout must travel along a path where:

- Each scout ensures that no town is visited more than once throughout their assignment.
- The two scouts' paths do not overlap at any town except the outpost.

The High Council wants to minimize the total number of towns visited by both scouts to ensure efficiency and reduce risk.

Your task: As the Royal Programmer, determine the minimum total number of towns visited by both scouts. If it is impossible for both scouts to reach the outpost under these constraints, report "Impossible".

Constraints

3 <= N <= 15

Input

The first line contains two space-separated integers:

N — the number of towns

M — the number of roads connecting them

The next **M** lines each contain two space-separated integers, **a** and **b**, indicating that town **a** and town **b** are connected by a road.

The following line contains two space-separated integers: the starting towns of the two scouts.

The last line contains the integer representing the outpost town.

Output

Output a single integer: the minimum number of towns visited by both scouts, or "Impossible" if such paths do not exist.

Time Limit (secs)

1

Examples

Example 1

input

14 16

1 2

2 3

3 4

4 5

5 6

6 7

1 8

8 9

9 10

10 7

1 11

11 12

11 13

13 14

14 7

10 11

1 12

7

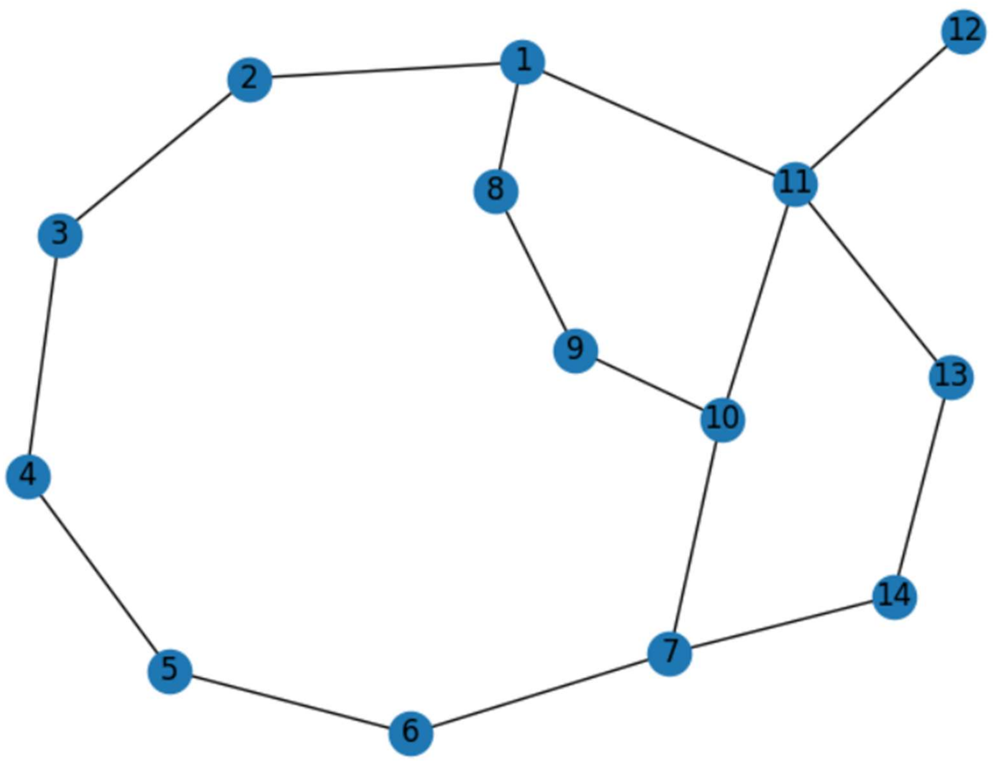
Output

8

Explanation

From the first line it is seen that there are 14 towns and 16 roads connecting them.

Arin is in town 1 and Bran is in town 12 and the outpost at 7.



Let us take paths taken by scouts as

Path taken by Arin: 1 -> 8 -> 9 -> 10 -> 7

Path taken by Bran: 12 -> 11 -> 13 -> 14 -> 7

The towns visited are: 1, 8, 9, 10, 11, 12, 13, 14 (8 towns in total). Hence 8 is the output.

Example 2

Input

8 9

1 2

2 3

3 4

4 9

5 9

3 6

6 5

1 8

8 6

1 2

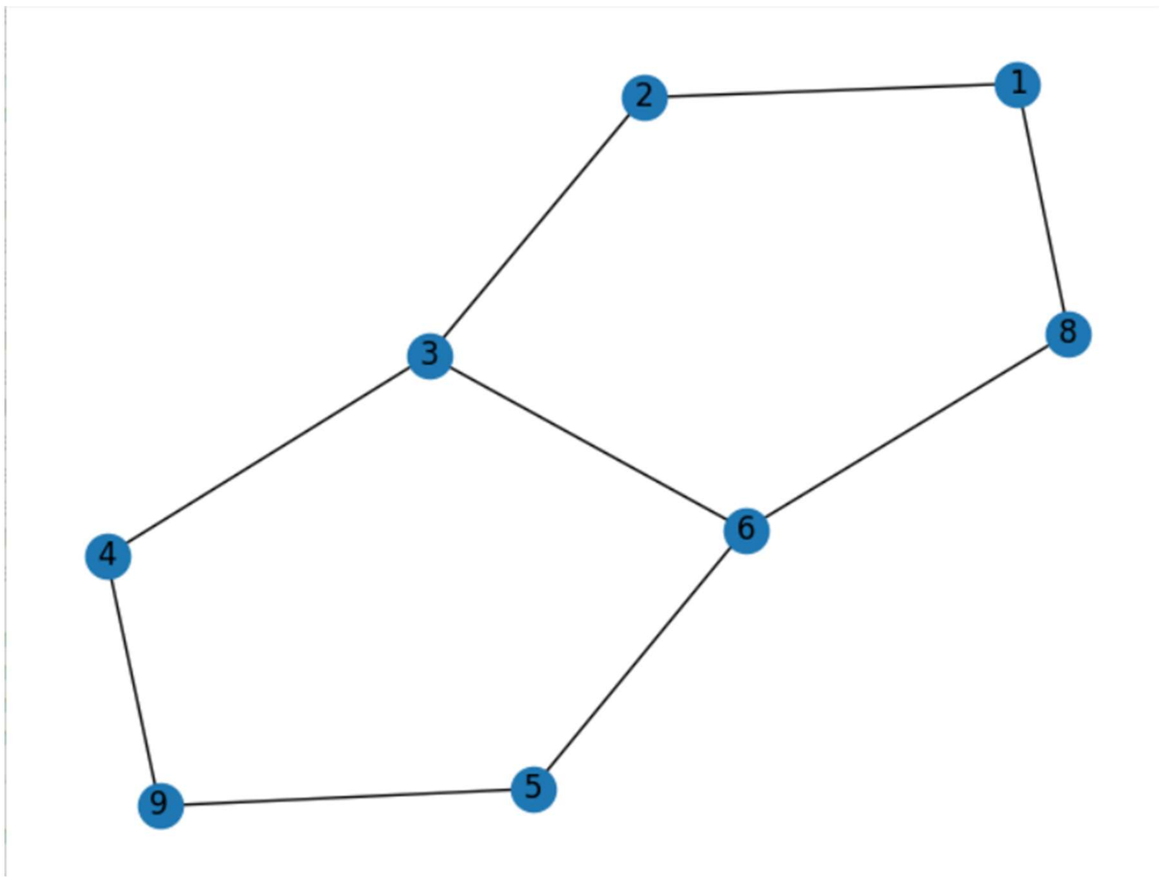
5

Output

7

Explanation

The scouts start at towns 1 and 2, with the outpost at town 5.



Let us take paths taken by scouts as

Arin's path: 1 -> 8 -> 6 -> 5

Bran's path: 2 -> 3 -> 4 -> 9 -> 5
The total number of unique towns visited is 7.

Stylus Simulator

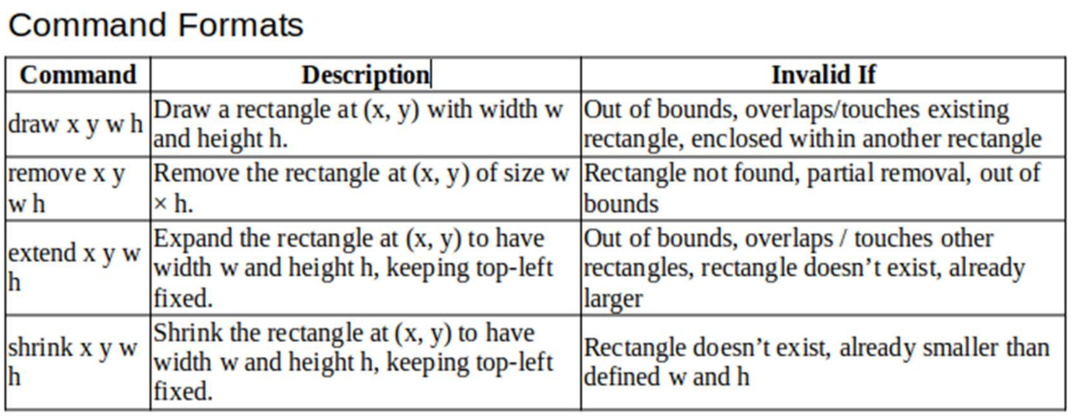
Problem Description

Raju, an avid digital artist, uses his tablet and stylus to create geometric art on a 2D canvas grid of size N × M. He carefully plans his drawing commands in advance and executes them sequentially, striving for perfection.

You are tasked with simulating Raju's drawing commands on a digital canvas. Each rectangle drawn is axis-aligned and defined by its top-left coordinates, width, and height.

Commands must be executed in the order provided. Invalid commands should be skipped but logged to the output. The following rules apply:

- Rectangles must remain fully within the grid.
- Rectangles must not touch or overlap each other, including at edges or corners.
- A rectangle completely enclosed within another is considered overlapping and is invalid.



Command Formats

Command	Description	Invalid If
draw x y w h	Draw a rectangle at (x, y) with width w and height h.	Out of bounds, overlaps/touches existing rectangle, enclosed within another rectangle
remove x y w h	Remove the rectangle at (x, y) of size w × h.	Rectangle not found, partial removal, out of bounds
extend x y w h	Expand the rectangle at (x, y) to have width w and height h, keeping top-left fixed.	Out of bounds, overlaps / touches other rectangles, rectangle doesn't exist, already larger
shrink x y w h	Shrink the rectangle at (x, y) to have width w and height h, keeping top-left fixed.	Rectangle doesn't exist, already smaller than defined w and h

Constraints

1<= M, N <= 20

1 <= C <= 20

1 <= w, h <= 20

The grid spans from (0,0) to (N, M).

For extend or shrink commands, if the rectangle sides are equal to those defined in the command, it is a valid command.

Input

The first line contains two integers, M and N, representing the height and width of the canvas grid, respectively.

The second line contains an integer C, denoting the number of commands to be executed.

Each of the following C lines describes a command in one of the formats above.

Output

Print each invalid command exactly as it appears in the input, in order.

After processing all commands, print the total number of valid rectangles remaining on the canvas.

Time Limit (secs)

1

Examples

Example 1

Input

8 6

7

draw 0 6 3 3

draw 4 3 2 2

shrink 0 6 2 2

remove 4 3 2 2

extend 0 6 7 7

shrink 0 6 4 4

shrink 0 6 1 1

Output

```
extend 0 6 7 7
shrink 0 6 4 4
1
```

Explanation

- The canvas is 6 × 8.
- The first rectangle is drawn at (0, 6) with size 3 × 3.
- The second rectangle is drawn at (4, 3) with size 2 × 2.
- The first rectangle shrunk to 2 × 2.
- The second rectangle is removed.
- The *extend* command is invalid (out of bounds).
- The *shrink* command is invalid (already smaller).
- The final shrink reduces the rectangle to 1 × 1.
- Only one rectangle remains.

Example 2

Input

```
4 4
5
draw 0 2 1 2
draw 2 4 2 2
shrink 0 2 1 1
extend 0 2 1 2
draw 2 2 1 1
```

Output

```
draw 2 2 1 1
2
```

Explanation

- The canvas is 4 × 4.
- The first rectangle is drawn at (0, 2) with size 1 × 2.

- The second rectangle is drawn at (2, 4) with size 2 × 2.
- The first rectangle shrunk to 1 × 1.
- The rectangle is extended back to 1 × 2.
- The third draw command will overlap existing rectangle, hence invalid.
- Two rectangles remain.

MahaKumbh 2K25

Problem Description

The Kumbh is a massive spiritual event held every 12 years in India, attracting millions of pilgrims. Every 12th Kumbh which comes after 144 years of previous Maha Kumbh is the most recent Maha Kumbh. Most recent such Maha Kumbh happened in 2025. This influx of devotees puts significant pressure on transportation services, especially the railways.

This grand spiritual event takes place at the sacred confluence of the Ganga, Yamuna, and the Saraswati rivers. During this time, millions of people travel from various nearby cities to participate in the holy festival, which is believed to cleanse the soul and wash away sins through a sacred dip at the confluence.

Due to the massive number of travellers, transport services around the festival site experience heavy rush and delays. Among all available modes of transport, the railways become the most preferred and widely used service by the devotees. As a result, railway stations and trains get extremely crowded, and managing the smooth movement of trains becomes a major challenge for the authorities.

To accommodate heightened demand, railway officials systematically oversee and adjust track connections throughout the day. Their main activities include:

1. Disconnecting certain tracks between stations to divert train routes and manage traffic.
2. Connecting tracks between stations to create new travel routes as needed.

Additionally, ongoing railway construction can restrict some trains from passing through specific stations. These restrictions apply only to trains starting from certain source stations.

Many families travel according to their own priorities and schedules. Before traveling, they need to check if a valid route exists from their starting station to their destination, considering the latest track arrangements and any restrictions.

Constraints

2 <= N, Q, R <= 14

Input

The first line contains a single integer, *N*, indicating the number of lines that define station connections.

Lines **2** to **N+1** show each source station and its connected stations as follows:

source_station, station1, station2 ... stationK (Connections are bidirectional.)

Line **N+2** contains single integer *Q* **-** representing the number of queries.

The following Q lines, i. e. lines **N+3** to **N+Q+2** contains *Q* queries in the below format.

- *source_station to destination_station*: Check if travel is possible between these stations.
- *station1 connects station2*: Connect these two stations.
- *station1 disconnects station2*: Disconnect these two stations.

The next line, Line **N+Q+3** contains a single integer, *R*, indicating the number of restriction lines.

The following R lines (**N+Q+4** to **N+Q+R+3**) specify restrictions in this format:

source_station, restricted_station1, restricted_station2.... restricted_stationM.

(Trains starting from source_station cannot pass through the listed restricted stations.)

Output

For each travel query, output "yes" or "no" (without quotes) on a separate line, indicating whether the journey is possible.

Time Limit (secs)

1

Examples

Example 1

Input

3

prayagraj varanasi chitrakoot

chitrakoot ayodhya lucknow vindhyachal

sarnath ayodhya chitrakoot kushinagar jaunpur

5

prayagraj to jaunpur

prayagraj connects kushinagar

prayagraj to jaunpur

prayagraj disconnects kushinagar

prayagraj to sarnath

2

chitrakoot jaunpur varanasi

prayagraj ayodhya

Output

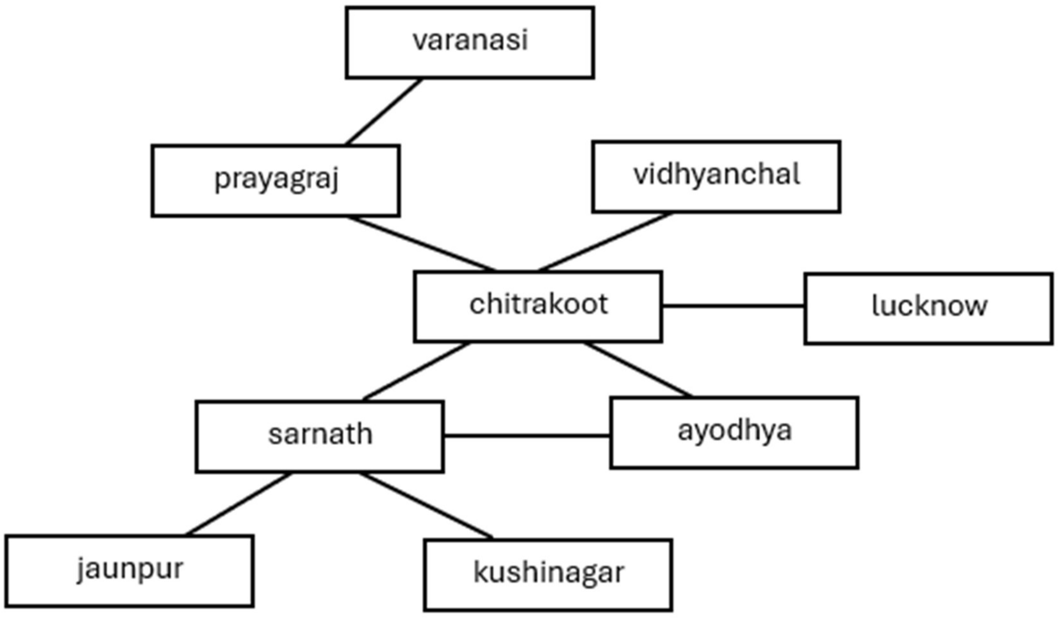
yes

yes

yes

Explanation

Initial connections between the stations are as follows:



When the first query is checked, the answer is visibly "yes", people can travel from *prayagraj* to *jaunpur*. There are multiple paths to go from *prayagraj* to *jaunpur* which includes one path containing *ayodhya*, but we must exclude that path considering the restrictions.

The second query states that the track between *prayagraj* and *kushinagar* gets connected. After this, when the third query is checked, the answer is "yes", people can travel from *prayagraj* to *jaunpur*.

The fourth query states that the track between *prayagraj* to *kushinagar* gets disconnected. After this, when the fifth query is checked, the answer remains "yes", people can travel through *prayagraj* to *sarnath*.

Example 2

Input

3

prayagraj varanasi chitrakoot

chitrakoot ayodhya lucknow vindhyachal

sarnath ayodhya kushinagar jaunpur

5

prayagraj to jaunpur

prayagraj connects kushinagar

prayagraj to jaunpur

prayagraj disconnects kushinagar

prayagraj to sarnath

2

chitrakoot jaunpur varanasi

prayagraj ayodhya

Output

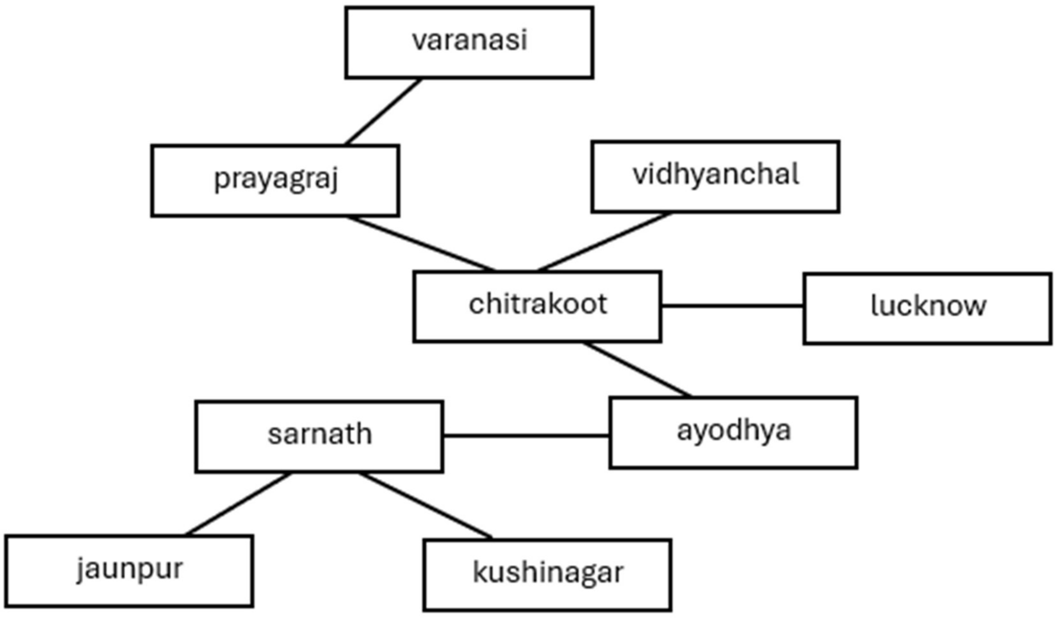
no

yes

no

Explanation

The initial connections between the stations are as follows:



When the first query is checked, the answer is "no" because there is only one path from *prayagraj* to *jaunpur* which includes visiting *ayodhya*. We must exclude that path because of the restrictions provided.

When the second query is executed, the track between *pyaragraj* to *kushinagar* is connected. Now when checking for the third query, the answer is "yes" as we have multiple paths going from *prayagraj* to *jaunpur*.

When the fourth query is executed, the path beween *prayagraj* and *kushinagar* is disconnected.

When checking for the fifth query, the answer is "no" because for going from *prayagraj* to *sarnath*, there is only one path, which passes through *ayodhya* which is restricted if the train's source station is *prayagraj*.

ReduceExp

Problem Description

Dr. Ada adjusted her glasses, focusing intently on the equations scrawled across the whiteboard. Her team was working on an artificial neural network, where each neuron evaluated a specific mathematical expression. They noticed that the network's performance was slow, and the culprit was the time it took to evaluate these expressions.

The computation time directly depended on the number of mathematical operations involved — primarily multiplication and addition.

Example: Consider the equation x × (1 + 2) + 1 + 3 × 1. This expression requires 5 operations to evaluate fully. However, by simplifying it to 3x + 4, the team reduced the required operations to just 2. Replacing complex equations with their simplified counterparts could significantly enhance the network's speed.

Further exploration revealed that certain quadratic equations could be factored without changing their results, further reducing the number of operations needed for evaluation. Ada's team identified three common forms amenable to this simplification:

- **Form 1:** a1a2xx + a1b2x + a2b1x + b1b2 can be simplified to (a1x + b1) (a2x + b2).
- **Form 2:** a1a2xy + a1b2x + a2b1y + b1b2 can be simplified to (a1x + b1) (a2y + b2).
- **Form 3:** a1a2xx + a1b2x + a2b1xy + b1b2y can be simplified to (a1x + b1y) (a2x + b2).

Here, *x* and *y* represent variables, while *a1, a2, b1,* and *b2* are constants. The key is matching the variable structure, not necessarily the exact equation coefficients. For instance, a1a2yy + a1b2y + a2b1y + b1b2 or a1a2yy + (a1b2+a2b1) y + b1b2 still fits Form 1.

The equation will contain only the variables x and y. The expression will never contain cubic terms, and it only have square terms (i.e., have x*x, x*y and not x*x*x, x*x*y etc.). Follow precedence with order of brackets, multiplication and then addition.

The Challenge: You are tasked with optimizing a given quadratic equation to minimize the number of operations required for evaluation. You can only apply multiplication, addition, and one of the three factoring forms mentioned above.

Good luck!

Note: You can apply only one form to the given equation to reduce it.

Constraints

Coefficients of the variables < 100

Input

A single-line equation involving the elements x,y,*,+,(,), and integers.

Output

Single integer representing minimum operations required after optimizing the equation.

Time Limit (secs)

1

Examples

Example 1

Input

x*(x+2+1)+1+1

Output

3

Explanation

We can do the usual reduction of multiplication and addition which will result in - xx+3x+2 and the number of operations required is 4.

After careful observation we can say that the equation is in first given form to factorize.

Applying it will give us - (x+1)*(x+2)

Above optimized equation requires only 3 operation which is the minimum possible. Hence print 3.

Example 2

Input

2*x*y+4*x+y+2+y*y

Output

6

Explanation

The given input equation requires 8 operations.

Now there is no further reduction by addiction and subtraction.

Factorizing it (2*x+1)*(y+2)+y*y now it requires 6 operation, which is the minimum. Hence print 6.

Example 3

Input

10*5+6*3

Output

0

Explanation

As we can see that the equation consists of only integers, and it can be evaluated fully to get the result 68.

Thus, the reduced one (68) have no operations to perform, hence print 0.

Activity Day

Problem Description

It is Activity Day at school, and the students are excited! The teacher has prepared a grid of boxes for a fun activity. The grid is of size N x M, with the bottom-left corner at coordinates (0, 0). Each box is rectangular and perfectly fits within the grid, defined by partition lines.

There are B boxes in total, and each of the B students is assigned a unique rank from 1 to B. Each student needs to select the box corresponding to their rank.

Ranking Criteria:

Boxes are ranked from 1 to B based on the following rules:

1. **Area:** Boxes with larger areas receive higher ranks.
2. **Bottom-Left Coordinates:** If two or more boxes have the same area, the box with the smaller x-coordinate of its bottom-left corner is ranked higher.
3. **Tiebreaker:** If the x-coordinates are also the same, the box with the smaller y-coordinate of its bottom-left corner is ranked higher.

Each box has a string written inside it. These strings are given in the input and placed in the boxes starting from (0,0), filling left to right and then bottom to top, one string per box. The value of a string is the sum of the ASCII values of its characters.

Raghu has been assigned rank R. Your task is to help him determine the value of the string inside the box he receives.

Constraints

1 <= N, M <= 100

1 <= number of partitioning lines <= 60

1 <= length of each string <= 10

The grid is perfectly filled by the B boxes and there won't be any empty spaces.

The number of strings is equal to the number of boxes.

Input

Line 1: Two integers, N and M, representing the dimensions of the grid.

Line 2: An integer L, representing the number of partition lines used to define the boxes.

The next L lines: Each line contains four space-separated integers x1 y1 x2 y2, representing the coordinates of a partition line segment. These line segments represent the internal partition lines that divide the grid into boxes.

Second Last Line: A single integer R, representing Raghu's assigned rank.

Last Line: B space-separated strings, representing the strings placed inside the boxes.

Output

Print a single integer denoting the value of the string Raghu receives.

Time Limit (secs)

1

Examples

Example 1

Input

4 4

5

0 1 4 1

2 1 2 3

0 2 2 2

3 0 3 1

0 3 4 3

2

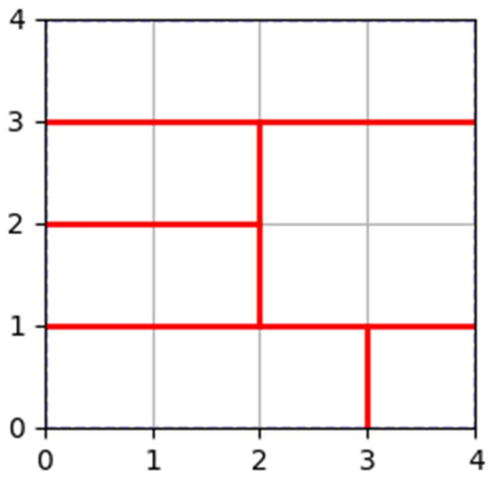
apple river table house dream cloud

Output

548

Explanation

When we visualize the above input, it looks like below.



The given input is shown in the above figure.

The input defines a 4x4 grid with 5 partition lines creating 6 boxes. The strings are assigned to the boxes in left-to-right, bottom-to-top order. We need to determine the string in the box ranked 2.

1. **Box Boundaries:** Based on the input, the box boundaries are:
   - Box 1: (0, 0) to (3, 1)
   - Box 2: (3, 0) to (4, 1)
   - Box 3: (0, 1) to (2, 2)
   - Box 4: (2, 1) to (4, 3)
   - Box 5: (0, 2) to (2, 3)
   - Box 6: (0, 3) to (4, 4)
2. **Area Calculation:** The area of each box is calculated:

- Box 1: 3 * 1 = 3
- Box 2: 1 * 1 = 1
- Box 3: 2 * 1 = 2
- Box 4: 2 * 2 = 4
- Box 5: 2 * 1 = 2
- Box 6: 4 * 1 = 4

Boxes 4 and 6 have the largest area, so they are candidates for Rank 1 and 2.

3. **Tiebreaker (Bottom-Left Coordinates):**
   - Box 4: (2, 1)
   - Box 6: (0, 3)

   Since Box 6 has a smaller x-coordinate, it receives Rank 1. Therefore, Box 4 receives Rank 2.

4. **String Assignment:** The boxes are assigned strings in left-to-right, bottom-to-top order. So, Box 4 contains the string "house".

5. **Calculate String Value:**

   The value of "house" is: h(104) + o(111) + u(117) + s(115) + e(101) = 548.

Therefore, the output is 548.

Example 2

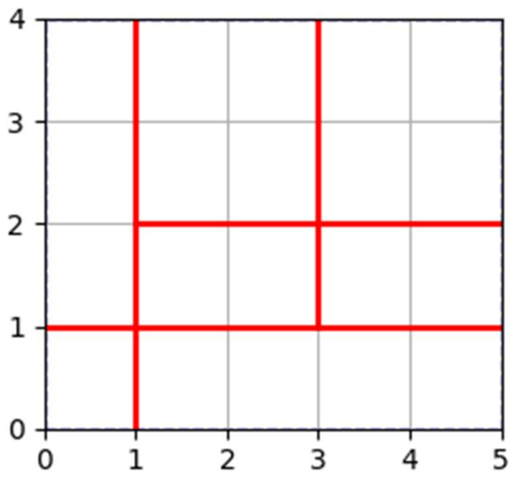
Input

```
5 4

4

1 2 5 2

1 0 1 4

3 1 3 4

0 1 5 1

4

laptop mouse keyboard desktop pendrive cpu joystick
```

Output

```
849
```

Explanation



The given input is shown in the above figure.

The input defines a 5x4 grid with 4 partition lines creating 7 boxes. The strings are assigned to the boxes in left-to-right, bottom-to-top order. We need to determine the string in the box ranked 4.

1. **Box Boundaries:** Based on the input, the box boundaries are:
   - Box 1: (0, 0) to (1, 1)
   - Box 2: (1, 0) to (5, 1)
   - Box 3: (0, 1) to (1, 4)
   - Box 4: (1, 1) to (3, 2)
   - Box 5: (3, 1) to (5, 2)
   - Box 6: (1, 2) to (3, 4)
   - Box 7: (3, 2) to (5, 4)
2. **Area Calculation:** The area of each box is calculated:
   - Box 1: 1 * 1 = 1
   - Box 2: 4 * 1 = 4
   - Box 3: 1 * 3 = 3
   - Box 4: 2 * 1 = 2
   - Box 5: 2 * 1 = 2
   - Box 6: 2 * 2 = 4
   - Box 7: 2 * 2 = 4

   Boxes 2, 6, 7 have the largest area of 4, and so they are candidates for Rank 1, 2, 3.
3. **Tiebreaker (Bottom-Left Coordinates):**
   - Box 2: (1, 0)
   - Box 6: (1, 2)
   - Box 7: (3, 2)

Box 2 and 6 has the smallest x-coordinates, they are candidates for Rank 1. In that case we will assign Rank 1 to which ever box is having the smallest y-coordinate. Since Box 2 is having smallest y-coordinate, it will be assigned Rank 1, box 6 will be assigned Rank 2 and finally box 7 will be assigned Rank 3.

As you can see, the candidate with Rank 4 is assigned to Box 3.

4. **String Assignment:** The boxes are assigned strings in left-to-right, bottom-to-top order. So, Box 3 (bottom left coordinates (0,1)) contains the string "keyboard".

5. **Calculate String Value:**

The value of "keyboard" is: k(107) + e(101) + y(121) + b(98) + o(111) + a(97) + r(114) + d(100) = 849.

Therefore, the output is 849.