Problem A: Intranet

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The large company you work for just wired its new fiber optic intranet. Since the process was pretty chaotic, they're not sure if they made any mistakes. Ideally, each two computers in the network should now be able to establish a connection through a series of fiber optic cables. Can you check whether this is the case?

Input

The input consists of

- one line containing N and M ($1 \leq N \leq 100, 1 \leq M \leq 10^4$) – the number of computers and cables, respectively
- M lines describing the cables, with the i-th line containing integers u_i and v_i ($1 \leq u_i, v_i \leq N$), denoting that there is a cable between computers u_i and v_i.

Output

Output "YES" if each two computers can establish a connection and "NO" otherwise.

Sample input and output

Input	Output
3 2 1 2 3 2	YES
3 1 1 3	NO

Problem B: Mine

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Recently you bought an archaic goldmine. Of course your main interest now is to find out exactly how deep beneath Earth's surface it leads.

The mine consists of a fairly large number of small plateaus that are connected by variously inclined tunnels. Apparently due to the mine being older than all reasonable mine safety rules, any two of the plateaus are only linked by precisely one path through the tunnels. From construction plans you bought with the mine you already know how much height each tunnel covers. Therefore you only need to combine all those pieces of information to calculate the total height.

Input

The input consists of

- one line containing N ($2 \leq N \leq 10^4$) – the number of plateaus (they are numerated from 1 to N with plateau 1 modeling the sole entrance into the mine)
- $N - 1$ lines describing the tunnels, with the i-th containing integers p_i and h_i ($1 \leq p_i \leq N, 1 \leq h_i \leq 1000$), denoting that there is a tunnel leading h_i meters downwards on its way from plateau p_i to plateau $(i + 1)$.

Output

Output a single integer – the height difference between the mine's lowest point and the entrance plateau (in meters).

Sample input and output

Input	Output
5 1 50 2 30 2 10 4 40	100

Problem C: Calculator

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Maths homework in grade school is boring. Seemingly endlessly you have to add and substract numbers... You are to write a program to save the poor children from that suffering.

Input

The input contains a mathematical expression of the following format:

- expr := integer
- expr := (expr)+(expr)
- expr := (expr)-(expr)

The string does not contain any whitespace and it is guaranteed, that the solution of any expression (also the intermediate results) fits into an int. The total length of the string does not exceed 10^6 characters.

Output

Output one integer: the solution of the expression.

Sample input and output

Input	**Output**
(7)-((1)+(2))	4
((5)+(9))-((4)-(1))	11

Problem D: Forest Love

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

As a Greenpeace activist you love forests very much. Unfortunately, the undirected graph you are given is not yet a forest. To change that, you want to remove some of its edges. Find out the minimum number of edges that need to be removed in order to make the graph a forest.

Input

The input consists of

- one line containing N and M ($1 \leq N \leq 10^5, 1 \leq M \leq 2 \cdot 10^5$) – the numbers of vertices and edges, respectively
- M lines each containing two integers u_i and v_i ($1 \leq u_i, v_i \leq N$), signifying an edge between vertices u_i and v_i.

Output

Output the minimum number of edges that need to be removed in order to make the graph a forest.

Sample input and output

Input	**Output**
5 4 1 2 2 3 3 1 5 4	1

Problem E: (p, q)-Knight

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 256 Mb

Problem description

In your spare time, you are developing a chess-based digital strategy game. At the moment you are in the process of incorporating several different kinds of (p, q)-knights into it. A (p, q)-knight is the generalized version of a normal chess knight, in each turn it moves p squares in one direction and q in the other. For instance, on a sufficiently large rectangular grid, a $(3, 4)$-knight can move from square $(5, 6)$ to the squares $(1, 3), (2, 2), (2, 10)$, $(1, 9), (8, 10), (9, 9), (8, 2)$ and $(9, 3)$. The common chess knight is obviously the $(2, 1)$-knight.

As you are not sure how strong which of them would be, you want to quantify their flexibility by a series of tests, each with the same basic premise: When placed on an $M \times N$ grid, at least how many moves does the knight need to get from one specified square to another one?

Input

The input consists of a single line containing 8 integers M, N, p, q, x_1, y_1, x_2 and y_2 ($1 \leq x_1, x_2 \leq M \leq 100, 1 \leq y_1, y_2 \leq N \leq 100, 0 \leq p, q \leq 100$).

Output

Output the minimal amount of moves a (p, q)-knight needs to reach square (x_2, y_2) from square (x_1, y_1). If this is impossible, output -1.

Sample input and output

Input	Output
3 3 1 1 1 1 3 3	2
2 2 1 1 1 1 1 2	-1

Problem F: Robots

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You are in charge of a number of robots in a factory hall. Right now they all sit on their respective docking stations, waiting for instructions. You were just informed that a specific piece of cargo currently located in your hall is urgently needed elsewhere, so you have to order one of your robots to pick it up and deliver it. To complete this task as fast as possible you need to find out which robot requires the least amount of actions to do the job and how many actions this are (only the last part has to be output).

The hall can be viewed as an $N \times M$ grid of quadratic cells, some of which are blocked by cargo, docking stations or machinery. For the purposes of this problem, your robots can only perform four (evenly costly) types of actions:

- Rotate 90° to the left
- Rotate 90° to the right
- Move 1 cell forward
- Pick up cargo

Specifically, the robots can rotate freely on any cell they're on without leaving it, but they can only move onto the cell that is adjacent to theirs in the direction they're facing and only if it's not blocked by anything. They can only pick up the cargo they must deliver and only when they're in a neighbouring cell (one with a common side with the cargo cell) and facing it.

Because the remaining delivery process is pretty much the same for any of the robots once they picked up the cargo, you are only interested in how many actions they need to do the first part (i.e. leave their docking station, find a way to the cargo and pick it up).

Input

The first line of input contains N and M ($1 \leq N, M \leq 100$) - the amounts of rows and columns the factory hall's grid consists of, respectively. The next N lines each contain M characters describing a row of the grid. The contents of the individual cells are coded as follows:

- '`_`' means the cell is free
- '`D`' means there is a docking station with a robot in it
- '`#`' means the cell is blocked by sth. else than a docking station
- '`C`' marks the cell with the requested cargo in it

Cells with docking stations in them can be seen as being almost the same as blocked cells, the only difference being that they are the possible starting points, i.e. it is possible to rotate on and leave a cell with a docking station in the beginning, but not to revisit it after that. Currently, all robots are oriented rightwards.

Output

Output a single integer, the least amount of actions any of your robots need to pick up the cargo. If no robot is able to do so, output `"IMPOSSIBLE"`.

Sample input and output

Input	**Output**
1 6 #D__C_	3
5 9 D_______ __###___# __#C___## __#___#D# _______#	10