Computer Programming Olympiad

A project of the Institute of IT Professionals South Africa

Ph: 021-448 7864 • Fax: 021-447 8410 • PO Box 13013, MOWBRAY, 7705 • info@olympiad.org.za • www.olympiad.org.za

Programming Olympiad 2020: Round 2

Not to be used before 19 August 2020

Make sure you understand everything in the "Rules & Instructions" tab of the contest before you start your timeframe.

1. The contest will open at 6 AM Wednesday 19 August and will shutdown at 6 PM.
2. This paper is only for invited Round 2 candidates. They will see the questions and submit programs on the saco-evaluator.
3. Any attempt to access any other website or source of information will disqualify you. Programs must pass the plagiarism check to be deemed valid.
4. Log on using the username and password given to you by your teacher.
5. There are four questions. There are 100 marks for each question. Each question is made up of subtasks. There are different marks for each subtask.
6. For each subtask, your score will be the highest score obtained across all your submissions for that question.
7. You will only be permitted to submit at most 30 submissions for the 2 hours of the contest.
8. You will have two hours (120 minutes) to complete all 4 questions.
9. You may assume that the user input will satisfy the problem specification and so you do not need to validate the input.
10. You may use the Communication pane (top left) but note that responses may be delayed.
11. You should not write code to produce only specific answers, as the judges will use other test cases.
12. Do not wait until the last few minutes to submit your programs.
13. After 120 minutes, or at 6:00 PM, whatever comes first, the server will end your participation. Should you finish before the time, check that you have answered all the questions.
14. Scores per question will be sent to your school after 19 August.

Question 1 - Count Special

Input file:	standard input
Output file:	standard output
Time limit:	0.1 seconds (subject to language multipliers)
Memory limit:	256 megabytes

Given two numbers *A* and *B*, we say a number is "special" if it is a positive integer and is divisible by either *A* or *B* (or both).

Write a program which will calculate how many special numbers there are below a given integer *N*.

Input

The input consists of 3 space-separated integers: *A*, *B*, and *N* ($1 \leq A,B \leq \min(N,10^6)$, $1 \leq N \leq 10^{18}$).

Output

Your program must output a single integer, the number of special numbers below *N*.

Scoring

Subtask 1 (0 points): Examples.

Subtask 2 (80 points): $1 \leq N \leq 10^6$.

Subtask 3 (20 points): $1 \leq N \leq 10^{18}$ (you may need to use the int 64 datatype).

Examples

standard input	standard output
2 3 10	6
4 6 50	16
12 21 999	119

Note

In the first example, we see 2, 3, 4, 6, 8 and 9 are all the numbers below 10 which are divisible by either 2 or 3. There are 6 of them, so we output the answer 6.

Question 2 - All about that Base

Input file:	standard input
Output file:	standard output
Time limit:	0.1 seconds (subject to language multipliers)
Memory limit:	256 megabytes

Our normal number system is base 10. So a number like 693 really represents $6 \times 10^2 + 9 \times 10^1 + 3 \times 10^0$. But we could instead use a base b ($2 \leq b \leq 36$). If $b > 10$, we use the digit *A* for the digit of value 10, *B* for 11, and so on. For example, 6AC1 (in base 14) represents $6 \times 14^3 + 10 \times 14^2 + 12 \times 14 + 1$ (in base 10).

Your task is, given the base b, and two numbers x and y in base b, compute their sum in base b.

Input

The input consists of 3 space-separated integers: b, x, and y. The values of x and y, if converted to base 10, are guaranteed to be less than 10^9.

Output

Output the sum of x and y, in base b.

Scoring

Subtask 1 (0 points): Examples.

Subtask 2 (50 points): $b = 7$.

Subtask 3 (50 points): $2 \leq b \leq 36$.

Examples

standard input	standard output
10 137 64	201
7 15 6	24
13 AB 12	C0

Question 3 - Grid Walk

Input file:	standard input
Output file:	standard output
Time limit:	0.2 seconds (subject to language multipliers)
Memory limit:	256 megabytes

You find yourself at the top-left cell of an N x M grid and would like to make your way to the bottom-right cell, through a sequence of steps. For each step, you are allowed to move either one cell to the right, or one cell down. However, you are not allowed to move into a "blocked" cell.

Write a program to determine the number of ways there are of reaching the bottom-right cell. Note, there might be 0 such ways of reaching the bottom-right cell. It is guaranteed that the top-left cell is not blocked.

Input

The first line contains two space-separated integers, N and M ($1 \leq N, M \leq 1000$). The next N lines contain M characters each, describing the rows of the grid. A **'.'** represents an empty cell, while a '#' represents a blocked cell.

Output

Output a single integer, the number of ways of reaching the bottom right cell of the grid. This number can be extremely large, so output only the remainder when divided by 10^8.

For example, if the answer was 1038000432, you would output 38000432. If the answer was 1700060120, you would output 60120.

Scoring

Subtask 1 (0 points): Examples.

Subtask 2 (10 points): $N = 1$.

Subtask 3 (20 points): $N = 2$.

Subtask 4 (20 points): $1 \leq N, M \leq 7$.

Subtask 5 (20 points): There are no blocked cells.

Subtask 6 (30 points): No further restrictions.

Examples

standard input	standard output
3 4 ....# #...	5
5 5 ...#. ....# .#... ....#	16
2 3 ..# ..#	0

Note

Let R indicate a step one cell to the right, and D indicate a step one cell down. Then in the first example, the different ways of reaching the bottom-right cell starting from the top-left are RRDDR, RDRDR, RDDRR, DRRDR and DRDRR.

Question 4 - Tree Slugs

Input file:	standard input
Output file:	standard output
Time limit:	1 second (subject to language multipliers)
Memory limit:	256 megabytes

K tree slugs are sitting on their favourite vertices of a tree. The tree consists of N vertices numbered from 1 to N connected by $N - 1$ edges, and each edge is initially covered in lichen.

After a fierce storm, each slug is blown to a new vertex. Now, the slugs want to make their way back to their favourite vertices, each following the shortest path to do so. Every time they walk along an edge, they will eat any lichen on that edge, leaving the edge lichen-free.

Calculate how many lichen-free edges there will be once all slugs are back at their favourite vertices.

Input

The first line contains N ($1 \leq N \leq 100\,000$) and K ($1 \leq K \leq 100\,000$), separated by a space.

The next $N-1$ lines each contain two integers u and v ($1 \leq u, v \leq N$) separated by a space, indicating that there is an edge between u and v.

The next K lines each contain two integers a and b ($1 \leq a, b \leq N$), indicating that a slug has favourite vertex a and was blown to the vertex b.

Output

Output a single integer, the number of lichen-free edges.

Scoring

Subtask 1 (0 points): Examples.

Subtask 2 (25 points): There is an edge between vertex i and $i+1$ for each i from 1 to $N-1$ (i.e. the tree is a single path).

Subtask 3 (25 points): $K = 1$.

Subtask 4 (20 points): All slugs are blown to the same vertex.

Subtask 5 (30 points): No further restrictions.

Examples (Please turn to page 6)

Examples

standard input	standard output
6 2 2 4 5 2 3 1 1 2 5 6 1 5 3 2	3
5 3 3 4 5 4 1 3 2 1 3 4 5 1 3 2	4
7 4 3 7 1 5 4 7 7 1 2 6 1 6 3 3 4 5 7 5 5 5	3

Note

In the first example, the tree is as shown below. The first slug makes their way from vertex 1 to vertex 5, traversing the edge 1-2 and the edge 2-5. The second slug makes their way from vertex 3 to vertex 2, traversing the edge 3-1 and the edge 1-2. So the 3 edges 1-2, 2-5 and 3-1 will all be lichen-free after this, and so we output 3.