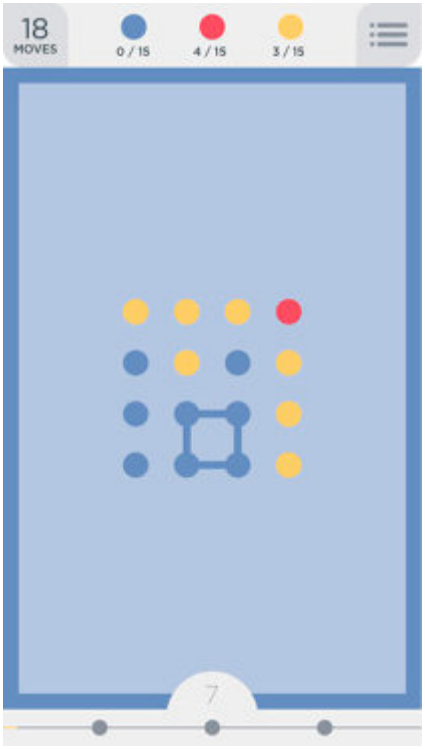
Problem A. Fox And Two Dots

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

Fox Ciel is playing a mobile puzzle game called "Two Dots". The basic levels are played on a board of size $n \times m$ cells, like this:



Each cell contains a dot that has some color. We will use different uppercase Latin characters to express different colors.

The key of this game is to find a cycle that contain dots of same color. Consider 4 blue dots on the picture forming a circle as an example. Formally, we call a sequence of dots $d_1, d_2, \ldots, d_k$ a *cycle* if and only if it meets the following condition:

1. These k dots are different: if $i \neq j$ then d_i is different from d_j.
2. k is at least 4.
3. All dots belong to the same color.
4. For all $1 \leq i \leq k-1$: d_i and d_{i+1} are adjacent. Also, d_k and d_1 should also be adjacent. Cells x and y are called adjacent if they share an edge.

Determine if there exists a *cycle* on the field.

Input

The first line contains two integers n and m ($2 \leq n, m \leq 50$): the number of rows and columns of the board.

Then n lines follow, each line contains a string consisting of m characters, expressing colors of dots in each line. Each character is an uppercase Latin letter.

Output

Output "`Yes`" if there exists a *cycle*, and "`No`" otherwise.

Examples

stdin	stdout
3 4 AAAA ABCA AAAA	Yes
3 4 AAAA ABCA AADA	No
4 4 YYYR BYBY BBBY BBBY	Yes
7 6 AAAAAB ABBBAB ABAAAB ABABBB ABAAAB ABBBAB AAAAAB	Yes
2 13 ABCDEFGHIJKLM NOPQRSTUVWXYZ	No

Note

In first sample test all '`A`' form a cycle.

In second sample there is no such cycle.

The third sample is displayed on the picture above ('`Y`' = Yellow, '`B`' = Blue, '`R`' = Red).

Problem B. Turtle

Input file:	`turtle.in`
Output file:	`turtle.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Turtle is living on a rectangular $n \times m$ grid. Each cell of a grid contains golden coins: there are a_{ij} coins in cell (i, j).

Initially turtle is located in cell $(1, 1)$. On each turn it can either move to cell $(i + 1, j)$ or $(i, j + 1)$, when located in cell (i, j).

	1	2	3	4	5
1	1	2	1	1	3
2	2	10	2	2	3
3	1	1	0	1	3

Turtle is collecting all the coins from the cells it visits. What is the maximum number of coins, that turtle can collect.

Input

The first line of input contains two integers: n and m ($1 \le n, m \le 100$) — the size of the grid.

Each of the next n lines contains m integers, describing the number of coins in grid cells: j-th number on i-th of these lines is equal to a_{ij} ($0 \le a_{ij} \le 100$).

Output

Output one integer: the maximum number of coins turtle can collect.

Examples

turtle.in	turtle.out
3 5 1 2 1 1 3 2 10 2 2 3 1 1 0 1 3	23
3 4 163 141 752 620 672 91 899 120 50 974 456 297	2708

Note

The best path in first sample testcase is:

	1	2	3	4	5
1	1				
2	2	10	2	2	3
3					3

Problem C. Time limit exceeded 8

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

```
#include <cstdio>

int f(int a, int b, int c) {
    if ((a & b) == (a | b)) return a + b + c;
    if ((a ^ b) != a)
        return f(a, b - 1, c + 1);
    if ((37 * a + b) % 46 < 13)
        return f(a + 1, b - 1, c);
    else
        return f(a + 2, b - 3, c + 1);
}


int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    printf("%d\n", f(n, m, 0));
}
```

Input

Output

Examples

standard input	standard output

Problem D. Yield

Input file:	`yield.in`
Output file:	`yield.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

You are given two real numbers a and b. Write a program to calculate $a + b$.

Input

The first line of the input file contains two real numbers — a and b ($-1000 \leq a, b \leq 1000$).

Output

Print the value of $a + b$ on the first line of the output file. The value must be precise up to four digits after the decimal point.

Examples

yield.in	yield.out
1.1 2.2	3.3
1 -1	0

Problem E. Zero-complexity Transposition

Input file:	`zero.in`
Output file:	`zero.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given a sequence of integer numbers. Zero-complexity transposition of the sequence is the reverse of this sequence. Your task is to write a program that prints zero-complexity transposition of the given sequence.

Input

The first line of the input file contains one integer n — length of the sequence ($0 < n \le 10\,000$). The second line contains n integer numbers — $a_1, a_2, \ldots, a_n$ ($-1\,000\,000\,000\,000\,000 \le a_i \le 1\,000\,000\,000\,000\,000$).

Output

On the first line of the output file print the sequence in the reverse order.

Examples

zero.in	zero.out
3 1 2 3	3 2 1
5 -3 4 6 -8 9	9 -8 6 4 -3