Sheet #3

A. Summation

2 seconds, 64 megabytes

Given a number N and an array A of N numbers. Print the **absolute summation** of these numbers.

absolute value : means to remove any negative sign in front of a number .

EX : |-5| = 5 , |7| = 7

Input

First line contains a number N $(1 \le N \le 10^5)$ number of elements.

Second line contains N numbers ($-10^9 \le A_i \le 10^9$).

Output

Print the **absolute summation** of these numbers.

input
4 7 2 1 3
output
13

input
3 -1 2 -3
output
2

Second Example :

-1 + 2 + -3 = -2 and it absolute is 2 so the answer is **2**.

B. Searching

2 seconds, 64 megabytes

Given a number N and an array A of N numbers. Determine if the number X **exists** in array A or **not** and print its position **(0-index)**.

Note: X may be found **once** or **more than once** and **may not be found**.

Input

First line contains a number N $(1 \le N \le 10^5)$ number of elements.

Second line contains N numbers $(0 \le A_i \le 10^9)$.

Third line contains a number X $(0 \le X \le 10^9)$.

Output

Print the **position** of X in the first time you find it. If it doesn't **exist** print **-1**.

input
3 3 0 1 0
output
1

input
5 1 3 0 4 5 10
output
-1

input
4 2 3 2 1 2
output
0

C. Replacement

1 second, 256 megabytes

Given a number N and an array A of N numbers. Print the array after doing the following operations:

- Replace every **positive** number by 1.
- Replace every **negative** number by 2.

Input
First line contains a number N $(2 \le N \le 1000)$ number of elements.

Second line contains N numbers ($-10^5 \le A_i \le 10^5$).

Output
Print the array after the **replacement** and it's values separated by space.

input
5 1 -2 0 3 4
output
1 2 0 1 1

D. Positions in array

1 second, 256 megabytes

Given a number N and an array A of N numbers. Print all array **positions** that store a number less than or equal to **10** and the **number stored** in that position.

Input
First line contains a number N $(2 \le N \le 1000)$ number of elements.

Second line contains N numbers ($-10^5 \le A_i \le 10^5$).

it's guaranteed that there is at least one number in array less than or equal to 10.

Output
For each number in the array that is equal to or less than **10** print a single line contains "A[i] = X", where **i** is the **position** in the array and X is the number **stored in the position**.

input
5 1 2 100 0 30
output
A[0] = 1 A[1] = 2 A[3] = 0

E. Lowest Number

1 second, 256 megabytes

Given a number N and an array A of N numbers. Print the **lowest number** and its **position**.

Note: if there are more than one answer print **first one's** position.

Input
First line contains a number N $(2 \le N \le 1000)$ number of elements.

Second line contains N numbers ($-10^5 \le A_i \le 10^5$).

Output
Print the **lowest number** and its **position (1-index)**.

input
3 1 2 3
output
1 1

input
5 5 6 2 3 2
output
2 3

F. Reversing

1 second, 64 megabytes

Given a number N and an array A of N numbers. Print the array in a **reversed order**.

Note:

*Don't use built-in-functions.

Input
First line contains a number N $(1 \le N \le 10^3)$ number of elements.

Second line contains N numbers $(0 \le A_i \le 10^9)$.

Output
Print the array in a **reversed order**.

input
4 5 1 3 2
output
2 3 1 5

input
5 1 2 3 4 5
output
5 4 3 2 1

G. Palindrome Array

1 second, 256 megabytes

Given a number N and an array A of N numbers. Determine if it's **palindrome** or **not**.

Note:

An array is called **palindrome** if it reads the same backward and forward, for example, arrays { **1** } and { **1,2,3,2,1** } are **palindromes**, while arrays { **1,12** } and { **4,7,5,4** } are **not**.

Input
First line contains a number N $(1 \le N \le 10^5)$ number of elements.

Second line contains N numbers $(1 \le A_i \le 10^9)$.

Output
Print "**YES**" (without quotes) if A is a **palindrome** array, otherwise, print "**NO**" (without quotes).

input
5 1 3 2 3 1
output
YES

input
3 3 1 2
output
1 2 3

input
4 1 2 3 4
output
NO

input
4 5 2 7 3
output
2 3 5 7

H. Sorting

1 second, 64 megabytes

Given a number N and an array A of N numbers. Print the numbers after **sorting** them.

Note:

- Don't use built-in-functions.
- try to solve it with bubble sort algorithm or Selection Sort.
- for more information watch : https://www.youtube.com/watch?v=EnodMqJuQEo.

Input

First line contains a number N ($0 < N < 10^3$) number of elements.

Second line contains N numbers $(-100 \le A_i \le 100)$.

Output

Print the numbers after **sorting** them.

I. Smallest Pair

1 second, 256 megabytes

Given a number N and an array A of N numbers. Print **the smallest** possible result of $A_i + A_j + \boldsymbol{j} - \boldsymbol{i}$, where $\mathbf{1} \le \mathbf{i} < \mathbf{j} \le \boldsymbol{N}$.

Input

The first line contains a number T $(1 \le T \le 100)$ number of test cases.

Each test case contains two lines:

- The first line consists a number N $(2 \le N \le 100)$ number of elements.
- The second line contains N numbers $(-10^6 \le A_i \le 10^6)$.

Output

For each test case print a single line contains **the smallest** possible sum for the corresponding test case.

input
1 4 20 1 9 4
output
7

First Case :

All possibles **(i,j)** where (**1** $\leq$ **i** < **j** $\leq$ N) are :

i = 1 , j = 2 then result = a_1 **+** a_2 **+ j - i = 20 + 1 + 2-1 = 22**.

i = 1 , j = 3 then result = a_1 **+** a_3 **+ j - i = 20 + 9 + 3-1 = 31**.

i = 1 , j = 4 then result = a_1 **+** a_4 **+ j - i = 20 + 4 + 4-1 = 27**.

i = 2 , j = 3 then result = a_2 **+** a_3 **+ j - i = 1 + 9 + 3-2 = 11**.

i = 2 , j = 4 then result = a_2 **+** a_4 **+ j - i = 1 + 4 + 4-2 = 7**.

i = 3 , j = 4 then result = a_3 **+** a_4 **+ j - i = 9 + 4 + 4-3 = 14**.

So the smallest possible result is **7**.

J. Lucky Array

1 second, 256 megabytes

Given a number N and an array A of N numbers. Determine if the array is **lucky** or **not**.

Note: the array is **lucky** if the **frequency** (number of occurrence) of the **minimum element** is **odd**.

Input

First line contains a number N $(2 \leq N \leq 1000)$ number of elements.

Second line contains N numbers $(-10^5 \leq A_i \leq 10^5)$.

Output

Print "**Lucky**" (without quotes) if the frequency of the **minimum element** is **odd**, otherwise print "**Unlucky**"(without quotes).

input
5 8 8 9 5 9
output
Lucky

input
5 3 3 3 5 3
output
Unlucky

First Example :

minimum element is **5** and its frequency is **1** and it's ODD so the array is **lucky**.

Second Example :

minimum element is **3** and its frequency is **4** and it's EVEN so the array is **not lucky**.

K. Sum Digits

2 seconds, 256 megabytes

Given a number N and an array A of N digits **(not separated by space)**. Print the **summation** of these digits.

Input

First line contains a number N $(1 \leq N \leq 10^6)$ number of digits.

Second line contains N digits $(0 \leq A_i \leq 9)$.

Output
Print the **summation** of these digits.

input
5 13305
output
12

First Example :

1 + 3 + 3 + 0 + 5 = 12 .

L. Max Subarray

1 second, 256 megabytes

A sub-array of array is an array composed from a contiguous block of the original array's elements.

In other words A sub-array **A[i-j]**, where $(1 \le i \le j \le N)$, is a sequence of integers $A_i, A_{i+1}, ..., A_j$.

For Example :

IF array = **[1,6,3,7]** then the **subarrays** are **[1] , [6] , [3] , [7] , [1,6] , [6,3], [3,7], [1,6,3] , [6,3,7] , [1,6,3,7]** .

Something like **[1,3]** would not be a sub-array as it's not a contiguous subsection of the original array.

Given a number N and an array A of N numbers. Print the **maximum** number of every sub-array separated by space.

Input
First line contains a number T $(1 \le T \le 5)$ number of test cases.

Each test case contains two lines:

- First line contains a number N $(1 \le N \le 100)$ number of elements.
- Second line contains N numbers $(-10^5 \le A_i \le 10^5)$.

Output
For each test case print a single line contains the **maximum** number of every sub-array separated by space.

print the answer in any order.

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

First Case :

All Sub arrays are :

[1] , [6] , [3] , [7] , [1,6] , [6,3],[3,7], [1,6,3] , [6,3,7] , [1,6,3,7]

- Sub-array **[1]** it maximum number is **1**.

- Sub-array **[6]** it maximum number is **6**.

- Sub-array **[3]** it maximum number is **3**.

- Sub-array **[7]** it maximum number is **7**.

- Sub-array **[1,6]** it maximum number is **6**.

- Sub-array **[6,3]** it maximum number is **6**.

- Sub-array **[3,7]** it maximum number is **7**.

- Sub-array **[1,6,3]** it maximum number is **6**.

- Sub-array **[6,3,7]** it maximum number is **7**.

- Sub-array **[1,6,3,7]** it maximum number is **7**.

so the maximum numbers are **[1,6,3,7,6,6,7,6,7,7] you can print them in any order**.

M. Replace MinMax

1 second, 256 megabytes

Given a number N and an array A of N numbers. Print the array after doing the following operations:

- Find **minimum** number in these numbers.
- Find **maximum** number in these numbers.
- Swap **minimum** number with **maximum** number.

Input
First line contains a number N $(2 \le N \le 1000)$ number of elements.

Second line contains N numbers $(-10^5 \le A_i \le 10^5)$

It's **guaranteed** that all numbers are distinct.

Output
Print the array after the **replacement** operation.

input
5 4 1 3 10 8
output
4 10 3 1 8

N. Check Code

1 second, 256 megabytes

Given two numbers A, B and a code S consisting of digits (**0,1,2,...,9**) and a symbol '-'.

Determine if the code follows the following rules or not:

- The **position** $A + 1$ in the code is the symbol '-'.
- All other characters are one of the following **digits**: (**0,1,2,...,9**).

Input
First line contains two numbers A, B $(1 \le A, B \le 10)$.

Second line contains S $(|S| = A + B + 1)$ and consists of '-' and digits from 0 through 9.

Output
Print "**Yes**" if the code S follows the above rules otherwise, print "**No**".

input
3 3 269-665
output
Yes

input
1 1 12-
output
No

input
1 2 7444
output
No

First example:

The (A+1)-th character of code is '-', and the other characters are digits from '**0**' through '**9**', so it follows the format.

O. Fibonacci

1 second, 256 megabytes

Given a number N. Print the **Fibonacci** number of N.

Note: In order to create the Fibonacci sequence use the following function:

- fib(1) = 0.
- fib(2) = 1.
- fib(n) = fib(n - 1) + fib(n - 2).

Input
Only one line containing a number N $(1 \leq N \leq 50)$.

Output
Print the **Fibonacci** number of N.

input
1
output
0

input
5
output
3

For more information visit Fibonacci: https://www.mathsisfun.com/numbers/fibonacci-sequence.html.

P. Minimize Number

1 second, 256 megabytes

Given a number N and an array A of N **positive** numbers. Print **maximum** possible operations that can be performed.

The operation is as follows: if all numbers are **even** then divide each of them by **2** otherwise, you can not perform any more operations.

Input
First line contains a number N $(1 \leq N \leq 200)$ number of elements.

Second line contains N numbers ($1 \leq A_i \leq 10^9$).

Output
Print the **maximum** possible number of operations that can be performed.

input
3 8 12 40
output
2

input
4 5 6 8 10
output
0

First example:

Initially, [**8,12,40**] are written on the blackboard. Since all those integers are even, You can perform the operation.

After the operation is performed once, [**4,6,20**] are written on the blackboard. Since all those integers are again even, You can perform the operation.

After the operation is performed twice, [**2,3,10**] are written on the blackboard. Now, there is an odd number **3** on the blackboard, so you cannot perform the operation any more.

Thus, you can perform the operation at most twice.

Second example:

Since there is an odd number **5** on the blackboard already in the beginning, You cannot perform the operation at all.

Q. Count Subarrays

1 second, 256 megabytes

A sub-array of array is an array composed from a contiguous block of the original array's elements.

In other words A sub-array **A[i-j]**, where $(1 \le i \le j \le N)$, is a sequence of integers $A_i, A_{i+1}, ..., A_j$.

For Example :

IF array = **[1,6,3,7]** then the **subarrays** are **[1] , [6] , [3] , [7] , [1,6] , [6,3], [3,7], [1,6,3] , [6,3,7] , [1,6,3,7]** .

Something like **[1,3]** would not be a sub-array as it's not a contiguous subsection of the original array.

Given a number N and an array A of N numbers. Print the number of sub-arrays which are **non-decreasing**.

Note:

- A sub-array **A[i-j]** is **non-decreasing** if **($A_i \le A_{i+1} \le A_{i+2} \le ... \le A_j$)**.

Input

First line contains a number $T\ (1 \le T \le 5)$ number of test cases.

Each test case contains two lines:

- First line contains a number $N\ (1 \le N \le 10^2)$ number of elements.
- Second line contains N numbers $(-10^5 \le A_i \le 10^5)$

Output

For each test case print a single line contains the number of sub-arrays which are **non-decreasing**..

input
2 5 1 4 2 3 5 1 5
output
9 1

First example:

All valid sub-arrays are :

- **[1] , [1,4] , [4] , [2] , [3] , [5] , [2,3] , [3,5] , [2,3,5]**

Second example:

Only single sub-array [**5**] is non-decreasing.

Note that singleton sub-arrays (have only one element) are identically non-decreasing.

R. Permutation with arrays

1 second, 256 megabytes

Given a number N and two arrays A, B of N numbers. Determine if B is a **permutation** of A or **not**.

Note: A **permutation** is an arrangement of all or part of a set of objects.

For example: The array [**2, 1, 3**], [**3, 2, 1**] and [**2, 3, 1**] are **permutation** of the array [**1, 2, 3**].

Input

First line contains a number N $(1 \leq N \leq 10^3)$ Number of elements.

Second line contains N numbers $(1 \leq A_i \leq 10^7)$ elements of array A.

Third line contains N numbers $(1 \leq B_i \leq 10^7)$ elements of array B.

Output

Print "**yes**" if array B is a permutation of A otherwise, print "**no**" without quotations.

input
4 4 2 3 7 2 3 4 9
output
no

input
5 5 1 1 9 3 1 9 1 5 3
output
yes

S. Search In Matrix

2 seconds, 64 megabytes

Given two numbers N and M, a 2D array of size $N * M$ and a number X. Determine whether X **exists** in the 2D array A or **not**.

Input

First line contains two numbers N, M $(2 \leq N, M \leq 100)$ N donates number of rows and M donates number of columns.

Each of the next N lines will contain M numbers $(1 \leq A_i \leq 10^5)$.

Last line contains a number X $(0 \leq X \leq 10^5)$ described above.

Output

Print "**will take number**" if the number **doesn't exist** in the 2D array otherwise, print "**will not take number**".

input
2 2 1 2 3 4 3
output
will not take number

input
2 2 1 2 3 4 10
output
will take number

T. Matrix

1 second, 256 megabytes

Given a number N and a 2D array A of size $N * N$. Print the **absolute difference** between the **summation** of its two diagonals (**primary diagonal and secondary diagonal**).

Input

First line contains a number N $(1 \leq N \leq 100)$ described above.

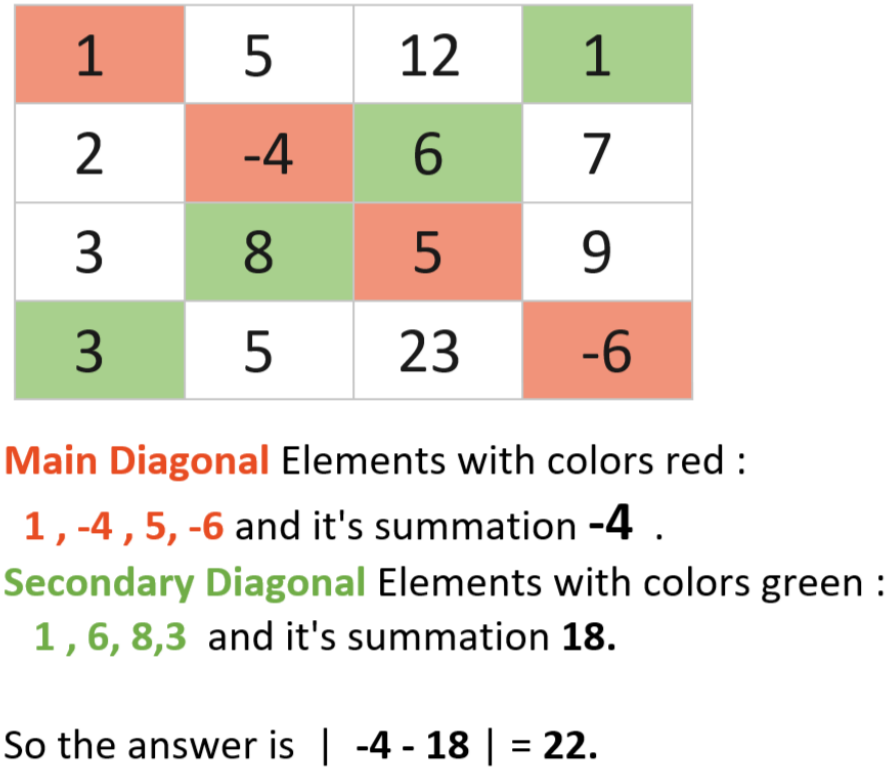
Each of the next N lines will contain N numbers ($-100 \leq A_i \leq 100$).

Output

Print the **absolute difference** between the **summation** of the matrix main diagonals.

input
4 1 5 12 1 2 -4 6 7 3 8 5 9 3 5 23 -6
output
22

First Example :



U. Is B a subsequence of A ?

1 second, 256 megabytes

a sub sequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements.

IF array = **[1,6,3 , 7]** then the **some subsequences** are **[1,3,7] , [6,7] , [1] , [6,3,7] , [1,7]** .

Something like **[3,1]** and **[6 , 7 , 1]** would not be sub sequences of the array **[1,6,3 , 7]**.

Given **2** numbers N, M and **2** arrays A consists of N numbers and B consists of M numbers. Determine whether B is a **sub-sequence** of A or **not**.

Note: The array B is called a **sub-sequence** of A if it's possible to **remove** zero or some elements from A to get B.

For example: if A=[**1,4,7**], and B is [**1**], [**1,4**], [**1,7**],[**1,4,7**] or [**4,7**] then B is a **sub-sequence** of A.

Input
First line contains two numbers N, M $(1 \leq N \leq 10^4, 1 \leq M \leq N)$, the sizes of arrays A and B respectively.

Second line contains N numbers $(1 \leq A_i \leq 10^9)$ elements of array A.

Third line contains M numbers $(1 \leq B_i \leq 10^9)$ elements of array B.

Output
Print "**YES**" (without the quotes), if B is a **sub-sequence** of A otherwise, print "**NO**" (without the quotes).

input
3 2 1 4 7 1 7
output
YES

input
7 4 1 8 4 7 5 2 7 4 5 7 2
output
NO

input
3 3 21 8 40 21 8 40
output
YES

V. Frequency Array

1 second, 256 megabytes

Given **2** numbers N, M and an array A of N numbers. For every number from **1** to M, print how many times this number **appears** in this array.

Input
First line contains two numbers N, M $(1 \leq N \leq 10^5, 1 \leq M \leq 10^5)$.

Second line contains N numbers $(1 \leq A_i \leq M)$.

Output
Print M lines, the i_{th} line should contain number of times that the number i appears in A

input
10 5 1 2 3 4 5 3 2 1 5 3
output
2 2 3 1 2

Numbers from 1 to 5 appearance are :

- 1 appears **2** times in the array .

- 2 appears **2** times in the array.
- 3 appears **3** times in the array.
- 4 appears **once** in the array.
- 5 appears **2** times in the array.

W. Mirror Array

1 second, 256 megabytes

Given two numbers N, M and a 2D array of size $N * M$. Print the **inverted** array that appeared in the mirror.

Input

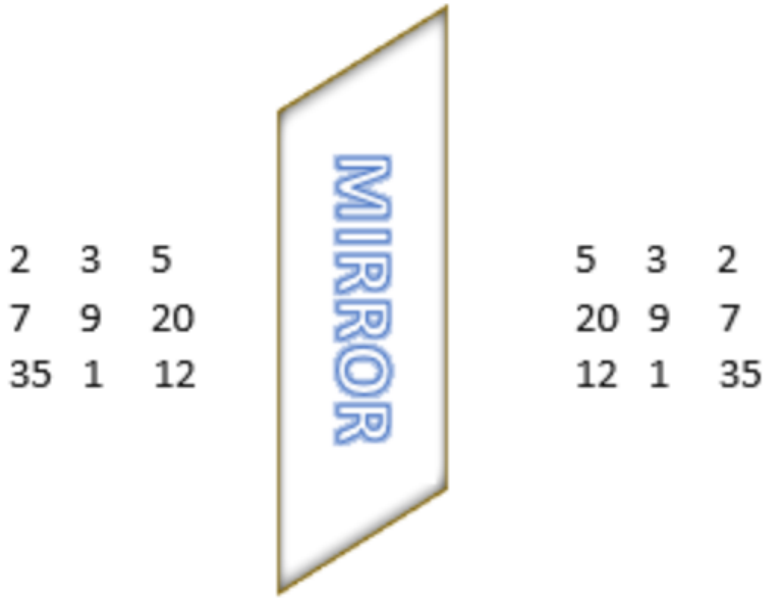
First line contains two numbers N, M $(1 \leq N, M \leq 100)$ N donates number of rows and M donates number of columns.

Each of the next N lines will contain M numbers $(1 \leq A_{i,j} \leq 10^9)$.

Output

Print the **inverted** array.

input
3 3 2 3 5 7 9 20 35 1 12
output
5 3 2 20 9 7 12 1 35



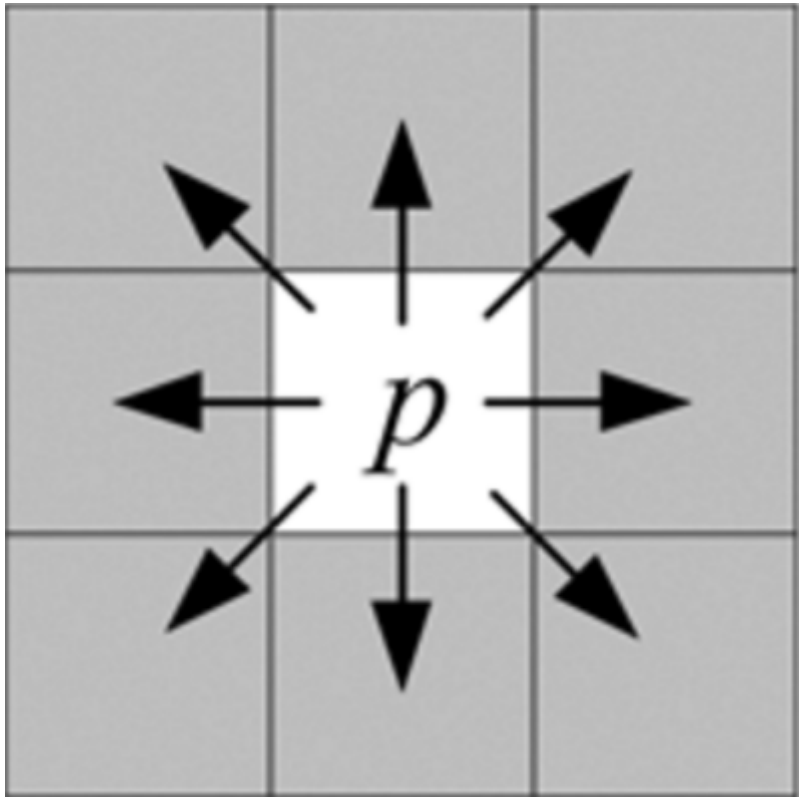
X. 8 Neighbors

1 second, 256 megabytes

Given two numbers N and M, a 2D array A of size $N * M$ which contains 'x' or '.' only and two numbers X, Y which donates a cell position in A such that X is the row number and Y is the column number.

Determine whether all neighbors of the given cell are 'x' or not.

Note: The neighbor cell is any cell that shares an **edge** or a **corner** and it should be **inside** 2D array.



Input

First line contains two numbers N, M $(2 \leq N, M \leq 100)$ N donates number of rows and M donates number of columns.

Each of the next N lines will contain M symbol can be ('.' or 'x').

Last line contains two numbers X, Y $(1 \leq X \leq N, 1 \leq Y \leq M)$.

Output

Print "**yes**" if all neighbors of the given cell are 'x' otherwise, print "**no**" without quotations.

input
3 3 xxx x.x xxx 2 2
output
yes

input
3 3 xxx xxx xx. 2 2
output
no

input
3 3 xxx xxx xxx 1 1
output
yes

Y. Range sum query

1.5 seconds, 256 megabytes

Given **2** numbers N and Q, an array A of N number and Q number of pairs L, R. For each query Q print a single line that contains the **summation** of all numbers from index L to index R.

Input
First line contains two numbers N, Q $(1 \leq N, Q \leq 10^5)$ where N is number of elements in A and Q is number of query pairs.

Second line contains N numbers$(1 \leq A_i \leq 10^9)$.

Next Q lines contains L, R $(1 \leq L \leq R \leq N)$.

Output
For each query Q print a single line that contains the **summation** of all numbers from index L to index R.

input
6 3 6 4 2 7 2 7 1 3 3 6 1 6

output
12 18 28

input
4 3 5 5 2 3 1 3 2 3 1 4

output
12 7 15

Z. Binary Search

1 second, 256 megabytes

Given 2 numbers N and Q, array A of N numbers and Q queries each one contains a number X.

For each query print a single line that contains **"found"** if the number X exists in array A otherwise, print **"not found"**.

Input
First line contains two numbers N, Q $(1 \leq N, Q \leq 10^5)$.

Second line contains N numbers $(1 \leq A_i \leq 10^9)$.

Next Q lines contains X $(1 \leq X \leq 10^9)$.

Output
Print the answer for each query in a single line.

input
5 3 1 5 4 3 2 5 3 6

output
found found not found