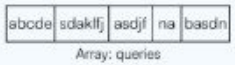


Array: queries

```
abcde
sdaklfj
asdjf
na
basdn
sdaklfj
asdjf
na
asdjf
na
basdn
sdaklfj
asdjf
5
abcde
sdaklfj
asdjf
na
basdn
```

Sample Output 3

```
1
3
4
3
2
```

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

- Sample Test case 0
- Sample Test case 1
- Sample Test case 2

```
7 aba
8 xzxb
9 ab
```

Your Output (stdout)

```
1 2
2 1
3 0
```

Expected Output — Download

```
1 2
2 1
3 0
```

Dynamic Array | HackerRank

hackerrank.com/challenges/dynamic-array/problem?isFullScreen=true

HackerRank | Prepare > Data Structures > Arrays > Dynamic Array

Exit Full Screen View

Submissions

Leaderboard

Discussions

Editorial

$arr[0]$ = [ ]

$arr[1]$ = [ ]

Query 0: Append 5 to $arr[(\ (0 \oplus 0)\ \%\ 2\ )] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5]

$arr[1]$ = [ ]

Query 1: Append 7 to $arr[(\ (1 \oplus 0)\ \%\ 2\ )] = arr[1]$.

$arr[0]$ = [5]

$arr[1]$ = [7]

Query 2: Append 3 to $arr[(\ (0 \oplus 0)\ \%\ 2\ )] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 3: Assign the value at index 0 of $arr[(\ (1 \oplus 0)\ \%\ 2\ )] = arr[1]$ to $lastAnswer$.

Store $lastAnswer$ in your answer array. $lastAnswer = 7$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 4: Assign the value at index 1 of $arr[(\ (1 \oplus 7)\ \%\ 2\ )] = arr[0]$ to $lastAnswer$.

Store $lastAnswer$ in your answer array. $lastAnswer = 3$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Return your answer array [7, 3]. The code stub prints its elements on separate lines.

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

```
2 1 0 5
3 1 1 7
4 1 0 3
5 2 1 0
6 2 1 1
```

Your Output (stdout)

```
1 7
2 3
```

Expected Output

Download

```
1 7
2 3
```

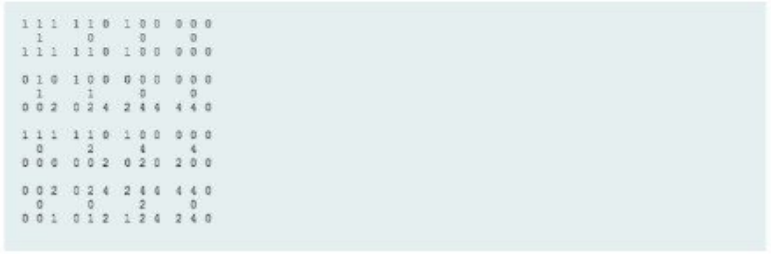
```
0 0 2 4 4 0
0 0 0 2 0 0
0 0 1 2 4 0
```

Sample Output

```
19
```

Explanation

arr contains the following hourglasses:



The hourglass with the maximum sum (19) is:

```
2 4 4
  2
1 2 4
```

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

- Sample Test case 0
- Sample Test case 1
- Sample Test case 2

Input (stdin) — Download

```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 2 4 4 0
0 0 0 2 0 0
0 0 1 2 4 0
```

Your Output (stdout)

```
19
```

Expected Output — Download

```
19
```

People connect with each other in a social network. A connection between Person i and Person j is represented as $\mathbf{M\ i\ j}$. When two persons belonging to different communities connect, the net effect is the merge the communities which i and j belong to.

At the beginning, there are n people representing n communities. Suppose person 1 and 2 connected and later 2 and 3 connected, then 1,2, and 3 will belong to the same community.

There are two types of queries:

1. $\mathbf{M\ i\ j} \implies$ communities containing persons i and j are merged if they belong to different communities.
2. $\mathbf{Q\ i} \implies$ print the size of the community to which person i belongs.

Input Format

The first line of input contains 2 space-separated integers n and q, the number of people and the number of queries.
The next q lines will contain the queries.

Constraints

$1 \leq n \leq 10^5$
$1 \leq q \leq 2 \times 10^5$

Output Format

The output of the queries.

Sample Input

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

```
7  Q 2
```

Your Output (stdout)

```
1  1
2  2
3  3
4  3
```

Expected Output — Download

```
1  1
2  2
3  3
4  3
```

An array is a data structure that stores elements of the same type in a contiguous block of memory. In an array, A, of size N, each memory location has some unique index, i (where $0 \leq i < N$), that can be referenced as $A[i]$ or A_i.

Your task is to reverse an array of integers.

Note: If you've already solved our C++ domain's Arrays Introduction challenge, you may want to skip this.

Example

$A = [1, 2, 3]$

Return $[3, 2, 1]$.

Function Description

Complete the function *reverseArray* with the following parameter(s):

- $int\ A[n]$: the array to reverse

Returns

- $int[n]$: the reversed array

Input Format

The first line contains an integer, N, the number of integers in A.

The second line contains N space-separated integers that make up A.

Constraints

- $1 \leq N \leq 10^3$
- $1 \leq A[i] \leq 10^4$, where $A[i]$ is the i^{th} integer in A

Copy Download

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
1 4
2 1 4 3 2
```

Your Output (stdout)

```
1 2 3 4 1
```

Expected Output Download

```
1 2 3 4 1
```

A *left rotation* operation on a circular array shifts each of the array's elements 1 unit to the left. The elements that fall off the left end reappear at the right end. Given an integer d, rotate the array that many steps to the left and return the result.

Example

$d = 2$

$arr = [1, 2, 3, 4, 5]$

After 2 rotations, $arr' = [3, 4, 5, 1, 2]$.

Function Description

Complete the *rotateLeft* function with the following parameters:

- *int* d: the amount to rotate by
- *int* $arr[n]$: the array to rotate

Returns

- *int*$[n]$: the rotated array

Input Format

The first line contains two space-separated integers that denote n, the number of integers, and d, the number of left rotations to perform.

The second line contains n space-separated integers that describe $arr[]$.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq d \leq n$
- $1 \leq a[i] \leq 10^6$

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
5 4
1 2 3 4 5
```

Your Output (stdout)

```
5 1 2 3 4
```

Expected Output Download

```
5 1 2 3 4
```

Complete the *preOrder* function in the editor below, which has 1 parameter: a pointer to the root of a binary tree. It must print the values in the tree's preorder traversal as a single line of space-separated values.

Input Format

Our test code passes the root node of a binary tree to the preOrder function.

Constraints

$1 \leq$ Nodes in the tree ≤ 500

Output Format

Print the tree's preorder traversal as a single line of space-separated values.

Sample Input

Explain

Sample Output

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Input (stdin) Download

```
6
1 2 5 3 6 4
```

Your Output (stdout)

```
1 2 5 3 4 6
```

Expected Output Download

```
1 2 5 3 4 6
```

A queue is an abstract data type that maintains the order in which elements were added to it, allowing the oldest elements to be removed from the front and new elements to be added to the rear. This is called a First-In-First-Out (FIFO) data structure because the first element added to the queue (i.e., the one that has been waiting the longest) is always the first one to be removed.

A basic queue has the following operations:

- Enqueue: add a new element to the end of the queue.
- Dequeue: remove the element from the front of the queue and return it.

In this challenge, you must first implement a queue using two stacks. Then process q queries, where each query is one of the following 3 types:

1. `1 x`: Enqueue element x into the end of the queue.
2. `2`: Dequeue the element at the front of the queue.
3. `3`: Print the element at the front of the queue.

Input Format

The first line contains a single integer, q, denoting the number of queries.
Each line i of the q subsequent lines contains a single query in the form described in the problem statement above. All three queries start with an integer denoting the query $type$, but only query 1 is followed by an additional space-separated value, x, denoting the value to be enqueued.

Constraints

- $1 \leq q \leq 10^5$
- $1 \leq type \leq 3$

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
10
1 42
2
1 14
3
1 28
3
1 60
1 78
2
2
```

Given a pointer to the root of a binary tree, print the top view of the binary tree.

The tree as seen from the top the nodes, is called the top view of the tree.

For example :

Top View : $1 -> 2 -> 5 -> 6$

Complete the function *topView* and print the resulting values on a single line separated by space.

Input Format

You are given a function,

```
void topView(node * root) {

}
```

Constraints

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
6
1 2 5 3 6 4
```

Your Output (stdout)

```
1 2 5 6
```

Expected Output Download

```
1 2 5 6
```

Constraints

- $3 \leq n \leq 10^7$
- $1 \leq m \leq 2 * 10^5$
- $1 \leq a \leq b \leq n$
- $0 \leq k \leq 10^9$

Sample Input

```
STDIN       Function
-----       --------
5 3         arr[] size n = 5, queries[] size q = 3
1 2 100     queries = [[1, 2, 100], [2, 5, 100], [3, 4, 100]]
2 5 100
3 4 100
```

Sample Output

```
200
```

Explanation

After the first update the list is `100 100 0 0 0`.

After the second update list is `100 200 100 100 100`.

After the third update list is `100 200 200 200 100`.

The maximum value is 200.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

- Sample Test case 0
- Sample Test case 1
- Sample Test case 2

Input (stdin) Download

```
5 3
1 2 100
2 5 100
3 4 100
```

Your Output (stdout)

```
200
```

Expected Output Download

```
200
```

There is a collection of input strings and a collection of query strings. For each query string, determine how many times it occurs in the list of input strings. Return an array of the results.

Example

$stringList = ['ab', 'ab', 'abc']$

$queries = ['ab', 'abc', 'bc']$

There are 2 instances of 'ab', 1 of 'abc', and 0 of 'bc'. For each query, add an element to the return array: $results = [2, 1, 0]$.

Function Description

Complete the function $matchingStrings$ with the following parameters:

- $string\ stringList[n]$: an array of strings to search
- $string\ queries[q]$: an array of query strings

Returns

- $int[q]$: the results of each query

Input Format

The first line contains and integer n, the size of $stringList[]$.
Each of the next n lines contains a string $stringList[i]$.
The next line contains q, the size of $queries[]$.
Each of the next q lines contains a string $queries[i]$.

Constraints

$1 \leq n \leq 1000$

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

- Sample Test case 0
- Sample Test case 1
- Sample Test case 2

```
4
aba
baba
aba
xzxb
3
aba
xzxb
ab
```

Your Output (stdout)

```
2
1
0
```

$arr[0]$ = [ ]

$arr[1]$ = [ ]

Query 0: Append 5 to $arr[(\ (0 \oplus 0)\ \%\ 2\ )] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5]

$arr[1]$ = [ ]

Query 1: Append 7 to $arr[(\ (1 \oplus 0)\ \%\ 2\ )] = arr[1]$.

$arr[0]$ = [5]

$arr[1]$ = [7]

Query 2: Append 3 to $arr[(\ (0 \oplus 0)\ \%\ 2\ )] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 3: Assign the value at index 0 of $arr[(\ (1 \oplus 0)\ \%\ 2\ )] = arr[1]$ to $lastAnswer$.
Store $lastAnswer$ in your answer array. $lastAnswer = 7$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 4: Assign the value at index 1 of $arr[(\ (1 \oplus 7)\ \%\ 2\ )] = arr[0]$ to $lastAnswer$.
Store $lastAnswer$ in your answer array. $lastAnswer = 3$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Return your answer array [7, 3]. The code stub prints its elements on separate lines.

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

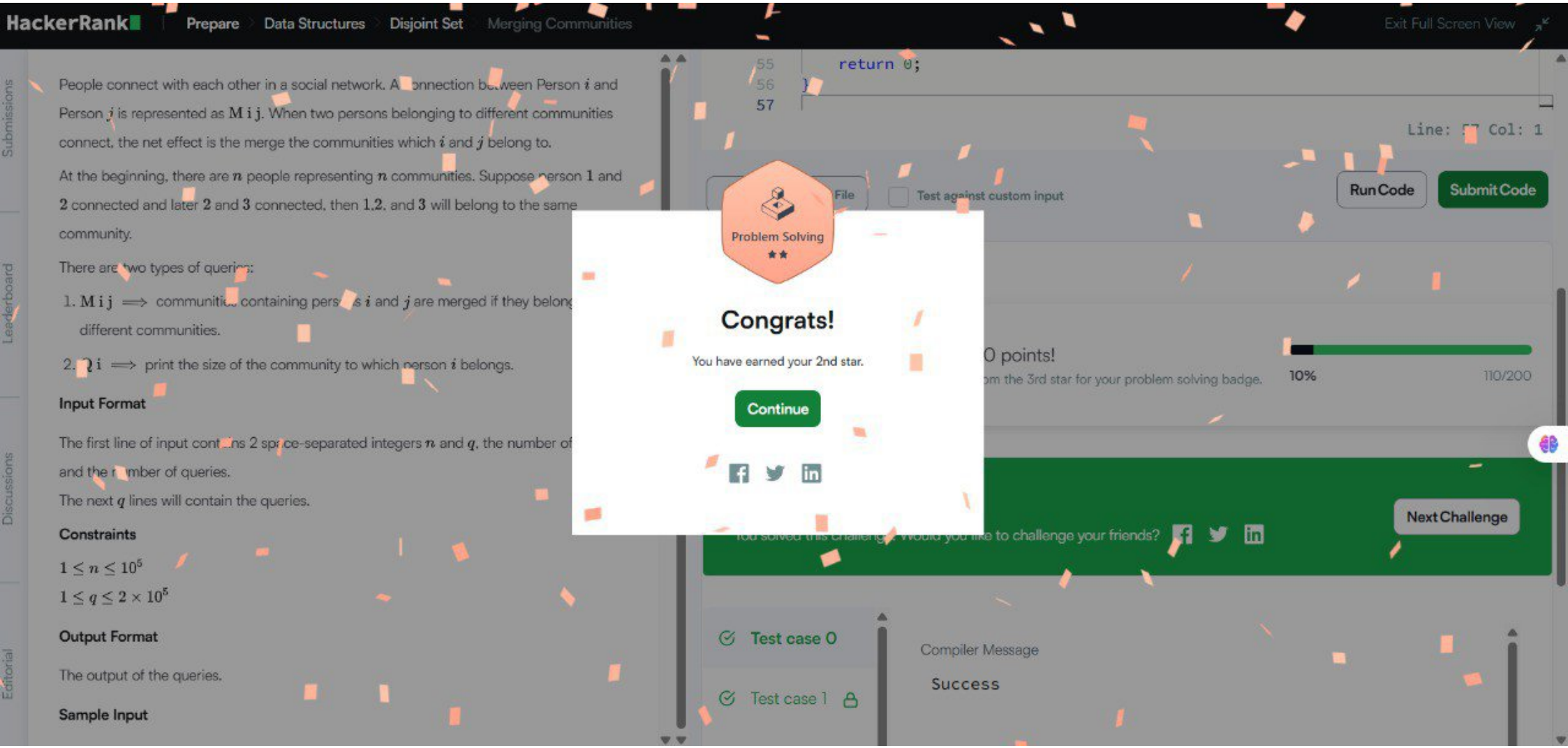
Input (stdin) Download

```
2 5
1 0 5
1 1 7
1 0 3
2 1 0
2 1 1
```

Your Output (stdout)

```
7
3
```

Expected Output Download


HackerRank
Prepare
Data Structures
Disjoint Set
Merging Communities
Exit Full Screen View
People connect with each other in a social network. A connection between Person i and Person j is represented as $\mathbf{M\ i\ j}$. When two persons belonging to different communities connect, the net effect is the merge the communities which i and j belong to.
At the beginning, there are n people representing n communities. Suppose person 1 and 2 connected and later 2 and 3 connected, then 1,2, and 3 will belong to the same community.
There are two types of queries:
1. $\mathbf{M\ i\ j}$ ⟹ communities containing persons i and j are merged if they belong to different communities.
2. $\mathbf{Q\ i}$ ⟹ print the size of the community to which person i belongs.
Input Format
The first line of input contains 2 space-separated integers n and q, the number of people and the number of queries.
The next q lines will contain the queries.
Constraints
$1 \leq n \leq 10^5$
$1 \leq q \leq 2 \times 10^5$
Output Format
The output of the queries.
Sample Input
Submissions
Leaderboard
Discussions
Editorial
return 0;
Line: 57 Col: 1
Test against custom input
Run Code
Submit Code
Problem Solving
Congrats!
You have earned your 2nd star.
Continue
0 points!
10%
110/200
Next Challenge
Test case 0
Test case 1
Compiler Message
Success