

# GE23131-Programming Using C-2024

## Quiz navigation


[Show one page at a time](#)
[Finish review](#)
Question **1**

Correct

Marked out of 1.00

☐ Flag question

<b>Status</b>	Finished
<b>Started</b>	Monday, 13 January 2025, 10:52 AM
<b>Completed</b>	Monday, 13 January 2025, 11:00 AM
<b>Duration</b>	8 mins 31 secs

Given an array of integers, reverse the given array in place using an index and loop rather than recursion.

### Example

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

Return the array `[5, 4, 2, 3, 1]` which is the reverse of the input array.

### Function Description

Complete the function `reverseArray` in the editor below.

`reverseArray` has the following parameter(s):

`int arr[n]`: an array of integers

Return

`int[n]`: the array in reverse order

### Constraints

$1 \leq n \leq 100$

$0 < arr[i] \leq 100$

### Input Format For Custom Testing

The first line contains an integer,  $n$ , the number of elements in `arr`.

Each line  $i$  of the  $n$  subsequent lines (where  $0 \leq i < n$ ) contains an integer, `arr[i]`.

### Sample Case 0

#### Sample Input For Custom Testing

5  
1  
3  
2  
4  
5

#### Sample Output

5  
4  
2  
3  
1

### Explanation

The input array is `[1, 3, 2, 4, 5]`, so the reverse of the input array is `[5, 4, 2, 3, 1]`.

### Sample Case 1

#### Sample Input For Custom Testing

4  
17  
10  
21  
45

#### Sample Output

45  
21  
10  
17

### Explanation

The input array is `[17, 10, 21, 45]`, so the reverse of the input array is `[45, 21, 10, 17]`.

	Test	Expected	Got	
	<pre>int arr[] = {1, 3, 2, 4, 5}; int result_count; int* result = reverseArray(5, arr, &amp;result_count); for (int i = 0; i &lt; result_count; i++)     printf("%d\n", *(result + i));</pre>	5 4 2 3 1	5 4 2 3 1	

Passed all tests!

Question **2**

Correct

Marked out of 1.00

☐ Flag question

An automated cutting machine is used to cut rods into segments. The cutting machine can only cut a rod of length *minLength* or more, and it can only make one cut at a time. Given the array *lengths[]* representing the lengths of each segment, determine if it is possible to make the necessary cuts using this machine. The machine can cut a rod of length *minLength* or more, and it can only make one cut at a time. The lengths already, in the order given.

Example

*n* = 3  
*lengths* = [4, 3, 2]  
*minLength* = 7

The rod is initially *sum(lengths)* = 4 + 3 + 2 = 9 units long. First cut off the segment of length 4. Then check that the length 7 rod can be cut into segments of lengths 3 and 2. Since 7 is greater than or equal to *minLength* = 7, the final cut can be made. Return "Possible".

Example

*n* = 3  
*lengths* = [4, 2, 3]  
*minLength* = 7

The rod is initially *sum(lengths)* = 4 + 2 + 3 = 9 units long. In this case, the initial cut can be on the rod of length 4. Regardless of the length of the first cut, the remaining piece will be shorter than *minLength*. Therefore, the cut cannot be made, the answer is "Impossible".

Function Description

Complete the function *cutThemAll* in the editor below.

*cutThemAll* has the following parameter(s):  
*int lengths[n]*: the lengths of the segments, in order  
*int minLength*: the minimum length the machine can accept

Returns

## Constraints

- $2 \leq n \leq 10^5$
- $1 \leq t \leq 10^9$
- $1 \leq \text{lengths}[i] \leq 10^9$
- The sum of the elements of *lengths* equals the uncut rod length.

## Input Format For Custom Testing

The first line contains an integer, *n*, the number of elements in *lengths*.

Each line *i* of the *n* subsequent lines (where  $0 \leq i < n$ ) contains an integer, *lengths*[*i*].

The next line contains an integer, *minLength*, the minimum length accepted by the machine.

## Sample Case 0

## Sample Input For Custom Testing

STDIN    Function

-----

4    →    lengths[] size n = 4

3    →    lengths[] = [3, 5, 4, 3]

5

4

3

9    →    minLength= 9

## Sample Output

Possible

## Explanation

The uncut rod is  $3 + 5 + 4 + 3 = 15$  units long. Cut the rod into lengths of  $3 + 5 + 4 = 12$  and piece into lengths 3 and  $5 + 4 = 9$ . The remaining segment is  $5 + 4 = 9$  units and that is long cut.

## Sample Case 1

## Sample Input For Custom Testing

STDIN    Function

-----

3    →    lengths[] size n = 3

5    →    lengths[] = [5, 6, 2]

6

2

12    →    minLength= 12

## Sample Output

Impossible

## Explanation

Answer: (penalty regime: 0 %)

Reset answer

	Test	Expected	Got	
	long lengths[] = {3, 5, 4, 3}; printf("%s", cutThemAll(4, lengths, 9))	Possible	Possible	
	long lengths[] = {5, 6, 2}; printf("%s", cutThemAll(3, lengths, 12))	Impossible	Impossible	

Passed all tests!

Save the state of the flags