GE23131-Programming Using C-2024

Quiz navigation

1 2

Show one page at a time

Finish review

Status	Finished
Started	Sunday, 12 January 2025, 10:35 PM
Completed	Sunday, 12 January 2025, 10:45 PM
Duration	9 mins 30 secs

Question **1**

Correct

Marked out of 1.00

Flag question

Given an array of integers, reverse the given array in place using an index and loop rather than a built-in function.

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*.

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

17

Explanation

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

Answer: (penalty regime: 0 %)

Reset answer

```
/*
 * Complete the 'reverseArray' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts INTEGER_ARRAY arr as parameter.
 */

/*
 * To return the integer array from the function, you should:
 *     - Store the size of the array to be returned in the result_
 *     - Allocate the array statically or dynamically
 *
 * For example,
 * int* return_integer_array_using_static_allocation(int* result_c
 *     *result_count = 5;
 *
 *     static int a[5] = {1, 2, 3, 4, 5};
 *
 *     return a;
 * }
 *
 * int* return_integer_array_using_dynamic_allocation(int* result_
 *     *result_count = 5;
 *
 *     int *a = malloc(5 * sizeof(int));
 *
 *     for (int i = 0; i < 5; i++) {
 *         *(a + i) = i + 1;
 *     }
 *
 *     return a;
```

```c
#include<stdio.h>
#include<stdlib.h>
int* reverseArray(int arr_count, int *arr, int *result_count) {
    int *result =(int*)malloc(arr_count*sizeof(int));
    if(result==NULL){
        return NULL;
    }
    for (int i=0;i<arr_count;i++)
    {
        result[i]=arr[arr_count-i-1];
    }
    *result_count=arr_count;
    return result;

}
```

	Test	Expected	Got	

✓	int arr[] = {1, 3, 2, 4, 5};
	int result_count;
	int* result = reverseArray(5, arr,
	&result_count);
	for (int i = 0; i < result_count; i++)
	printf("%d\n", *(result + i));
		5
		4
		2
		3
		1
			5
			4
			2
			3
			1
				✓

Passed all tests! ✓

Question **2**

Correct

An automated cutting machine is used to cut rods into segments. The cutting machine can only hold a rod of *minLength* or more, and it can only make one cut at a time. Given the array *lengths[]* representing the desired lengths of each segment, determine

Flag question

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 + 3 = 7* leaving a rod *9 - 7 = 2*. Then check that the length *7* rod can be cut into segments of lengths *4* and *3*. 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 of length *4* or *4 + 2 = 6*. Regardless of the length of the first cut, the remaining piece will be shorter than *minLength*. Because *n - 1 = 2* cuts cannot be made, the answer is "*Impossible*".

Function Description

Complete the function *cutThemAll* in the editor below.

int lengths[n]: the lengths of the segments, in order

int minLength: the minimum length the machine can accept

Returns

string: "*Possible*" if all *n-1* cuts can be made. Otherwise, return the string "*Impossible*".

Constraints

- $2 \le n \le 10^5$
- $1 \le t \le 10^9$
- $1 \le lengths[i] \le 10^9$
- *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 $0 \le i < n$) contains an integer, *lengths[i]*.

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

Sample Case 0

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 *3*. Then cut the *12* unit piece into lengths *3* and *5 + 4 = 9*. The remaining segment is *5 + 4 = 9* units and that is long enough to make the final cut.

Sample Case 1

Sample Input For Custom Testing

STDIN Function

----- --------

3 → lengths[] size n = 3

2

12 → minLength= 12

Sample Output

Impossible

Explanation

The uncut rod is *5 + 6 + 2 = 13* units long. After making either cut, the rod will be too short to make the second cut.

Answer: (penalty regime: 0 %)

Reset answer

```
/*
 * Complete the 'cutThemAll' function below.
 *
 * The function is expected to return a STRING.
 * The function accepts following parameters:
 *  1. LONG_INTEGER_ARRAY lengths
 *  2. LONG_INTEGER minLength
 */

/*
 * To return the string from the function, you should either do st
 *
 * For example,
 * char* return_string_using_static_allocation() {
 *     static char s[] = "static allocation of string";
 *
 *     return s;
```

```
 *     char* s = malloc(100 * sizeof(char));
 *
 *     s = "dynamic allocation of string";
 *
 *     return s;
 * }
 *
 */
#include<stdio.h>
char* cutThemAll(int lengths_count, long *lengths, long minLength)
    long t=0,i=1;
    for(int i=0;i<=lengths_count-1;i++){
        t+=lengths[i];
    }
    do{
        if(t-lengths[lengths_count-1]<minLength){
            return "Impossible";


        }
        i++;
    }while(i<lengths_count-1);
    return "Possible";

}

```

	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	✓

Finish review