GE23131-Programming Using C-2024

Status	Finished
Started	Thursday, 16 January 2025, 12:26 PM
Completed	Thursday, 16 January 2025, 12:32 PM
Duration	6 mins 9 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 rath

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 ≤ n ≤ 100

0 < arr[i] ≤ 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 ≤ 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].

```
* int* return_integer_array_using_dynamic_allocation(int* result_count) {
 *     *result_count = 5;
 *
 *     int *a = malloc(5 * sizeof(int));
 *
 *     for (int i = 0; i < 5; i++) {
 *         *(a + i) = i + 1;
 *     }
 *
 *     return a;
 * }
 *
 */
int* reverseArray(int arr_count, int *arr, int *result_count) {
*result_count=arr_count;
int*result=(int*)malloc(arr_count* sizeof(int));
for(int i=0;i<arr_count;i++)
{
   result[i]=arr[arr_count-1-i];
}
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
Marked out of 1.00
Flag question

An automated cutting machine is used to cut rods into segments. The cutting machine c
only make one cut at a time. Given the array *lengths[]* representing the desired lengths c
necessary cuts using this machine. The rod is marked into lengths already, in the order g

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 le
the length *7* rod can be cut into segments of lengths *4* and *3*. Since *7* is greater than or e
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
the first cut, the remaining piece will be shorter than *minLength*. Because *n - 1 = 2* cuts c

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

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

- *$2 \le n \le 10^5$*
- *$1 \le t \le 10^9$*
- *$1 \le lengths[i] \le 10^9$*
- *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 mach

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*
4 = 9. The remaining segment is *5 + 4 = 9* units and that is long enough to make the fin

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

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

```
char* cutThemAll(int lengths_count, long *lengths, long minLength) {
int s=0;
for (int i=0;i<lengths_count-1;i++)
{
    s+=*(lengths+i);
}
if(s>=minLength){
    return "Possible";
}
else{
    return "Impossible";
```

	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