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 ≤ n ≤ 100

Answer: (penalty regime: 0 %)

Reset answer

```c
/*
 * 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_count variable
 *      - Allocate the array statically or dynamically
 *
 * For example,
 * int* return_integer_array_using_static_allocation(int* result_count) {
 *     *result_count = 5;
 *
 *     static int a[5] = {1, 2, 3, 4, 5};
 *
 *     return a;
 * }
 *
 * 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) {
    int* reverseArr=malloc(arr_count*sizeof(int));
    if(reverseArr==NULL){
        *result_count=0;
        return NULL;
    }
    for(int i=0;i<arr_count;i++){
        reverseArr[i]=arr[arr_count-1-i];
    }
    *result_count=arr_count;
    return reverseArr;
}
```

	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 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 if it is possible to make the necessary cuts using this machine. The rod is marked into lengths already, in the order given.

Answer: (penalty regime: 0 %)

Reset answer

```c
/*
 * 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 static allocation or dynamic allocation
 *
 * For example,
 * char* return_string_using_static_allocation() {
 *     static char s[] = "static allocation of string";
 *
 *     return s;
 * }
 *
 * char* return_string_using_dynamic_allocation() {
 *     char* s = malloc(100 * sizeof(char));
 *
 *     s = "dynamic allocation of string";
 *
 *     return s;
 * }
 *
 */
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char* cutThemAll(int lengths_count, long *lengths, long minlength) {
    long totallength=0;
    for(int i=0;i<lengths_count;i++){
        totallength+=lengths[i];
    }
    for(int i=0;i<lengths_count;i++){
        totallength-=lengths[i];
        if(totallength>=minlength){
            long remaininglength=totallength-minlength;
            if(remaininglength<=minlength){
                char *result=(char*)malloc(9*sizeof(char));
                strcpy(result,"Possible");
                return result;
            }
        }
        totallength+=lengths[i];
    }
    char *result=(char*)malloc(11*sizeof(char));
    strcpy(result,"Impossible");
    return result;
```