```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) {
    *result_count=arr_count;
    static int rev[100];
    int i,j=0;
    for(i=arr_count-1;i>=0;i--)
    rev[j++]=arr[i];
    return rev;

}
```

	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! ✓

```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;
 * }
 *
 */
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! ✓