# PROBLEM STATEMENT

HackerRank Practice > Data Structures > Arrays > Arrays - DS

Problem | Submissions | Leaderboard | Discussions | Editorial

An array is a type of data structure that stores elements of the same type in a contiguous block of memory. In an array, $A$, of size $N$, each memory location has some unique index, $i$ (where $0 \leq i < N$), that can be referenced as $A[i]$ or $A_i$.

Reverse an array of integers.

**Note:** If you've already solved our C++ domain's Arrays Introduction challenge, you may want to skip this.

**Example**

$A = [1, 2, 3]$

Return $[3, 2, 1]$.

**Function Description**

Complete the function reverseArray in the editor below.

reverseArray has the following parameter(s):

- int A[n]: the array to reverse

**Returns**

- int[n]: the reversed array

**Input Format**

The first line contains an integer, $N$, the number of integers in $A$.
The second line contains $N$ space-separated integers that make up $A$.

**Constraints**

- $1 \leq N \leq 10^3$
- $1 \leq A[i] \leq 10^4$, where $A[i]$ is the $i^{th}$ integer in $A$

**Sample Input 1**

Copy Download

| 1 | 4 | 3 | 2 |
|---|---|---|---|

Array: arr

```
4
1 4 3 2
```

**Sample Output 1**

```
2 3 4 1
```

## PROGRAM USED TO SOLVE THE PROBLEM STATEMENT

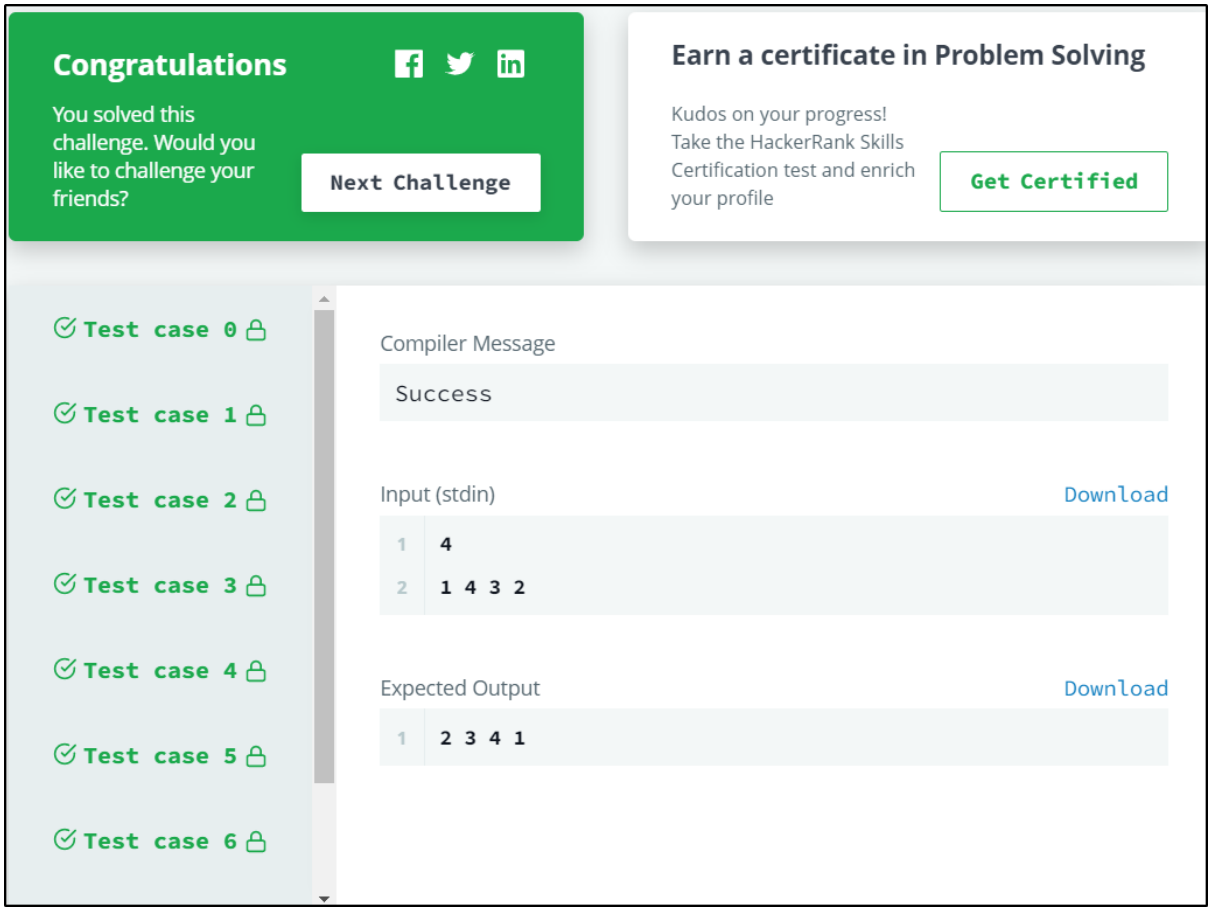
```
#include <assert.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char** split_string(char*);

// Complete the reverseArray function below.

// Please store the size of the integer array to be return
ed in result_count pointer. For example,
// int a[3] = {1, 2, 3};
//
// *result_count = 3;
//
// return a;
//
int* reverseArray(int a_count, int* a, int* result_count)
{
    *result_count = a_count;
    int temp;
    int end = a_count-1;
    for(int i = 0; i < a_count/2; i++){
        temp = *(a+end);
        *(a+end) = *(a+i);
        *(a+i) = temp;

        end--;
    }

    return a;
}

int main()
```

```
{
    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

    char* arr_count_endptr;
    char* arr_count_str = readline();
    int arr_count = strtol(arr_count_str, &arr_count_endpt
r, 10);

    if (arr_count_endptr == arr_count_str || *arr_count_en
dptr != '\0') { exit(EXIT_FAILURE); }

    char** arr_temp = split_string(readline());

    int* arr = malloc(arr_count * sizeof(int));

    for (int i = 0; i < arr_count; i++) {
        char* arr_item_endptr;
        char* arr_item_str = *(arr_temp + i);
        int arr_item = strtol(arr_item_str, &arr_item_endp
tr, 10);

        if (arr_item_endptr == arr_item_str || *arr_item_e
ndptr != '\0') { exit(EXIT_FAILURE); }

        *(arr + i) = arr_item;
    }

    int res_count;
    int* res = reverseArray(arr_count, arr, &res_count);

    for (int i = 0; i < res_count; i++) {
        fprintf(fptr, "%d", *(res + i));

        if (i != res_count - 1) {
            fprintf(fptr, " ");
        }
    }

    fprintf(fptr, "\n");

    fclose(fptr);
```

```
    return 0;
}

char* readline() {
    size_t alloc_length = 1024;
    size_t data_length = 0;
    char* data = malloc(alloc_length);

    while (true) {
        char* cursor = data + data_length;
        char* line = fgets(cursor, alloc_length - data_len
gth, stdin);

        if (!line) {
            break;
        }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_le
ngth - 1] == '\n') {
            break;
        }

        alloc_length <<= 1;

        data = realloc(data, alloc_length);

        if (!line) {
            break;
        }
    }

    if (data[data_length - 1] == '\n') {
        data[data_length - 1] = '\0';

        data = realloc(data, data_length);
    } else {
        data = realloc(data, data_length + 1);

        data[data_length] = '\0';
    }
```

```
    return data;
}

char** split_string(char* str) {
    char** splits = NULL;
    char* token = strtok(str, " ");

    int spaces = 0;

    while (token) {
        splits = realloc(splits, sizeof(char*) * ++spaces)
;

        if (!splits) {
            return splits;
        }

        splits[spaces - 1] = token;

        token = strtok(NULL, " ");
    }

    return splits;
}
```

# TEST CASES


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Earn a certificate in Problem Solving
Kudos on your progress! Take the HackerRank Skills Certification test and enrich your profile
Get Certified
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
4
1 4 3 2
Expected Output
Download
2 3 4 1