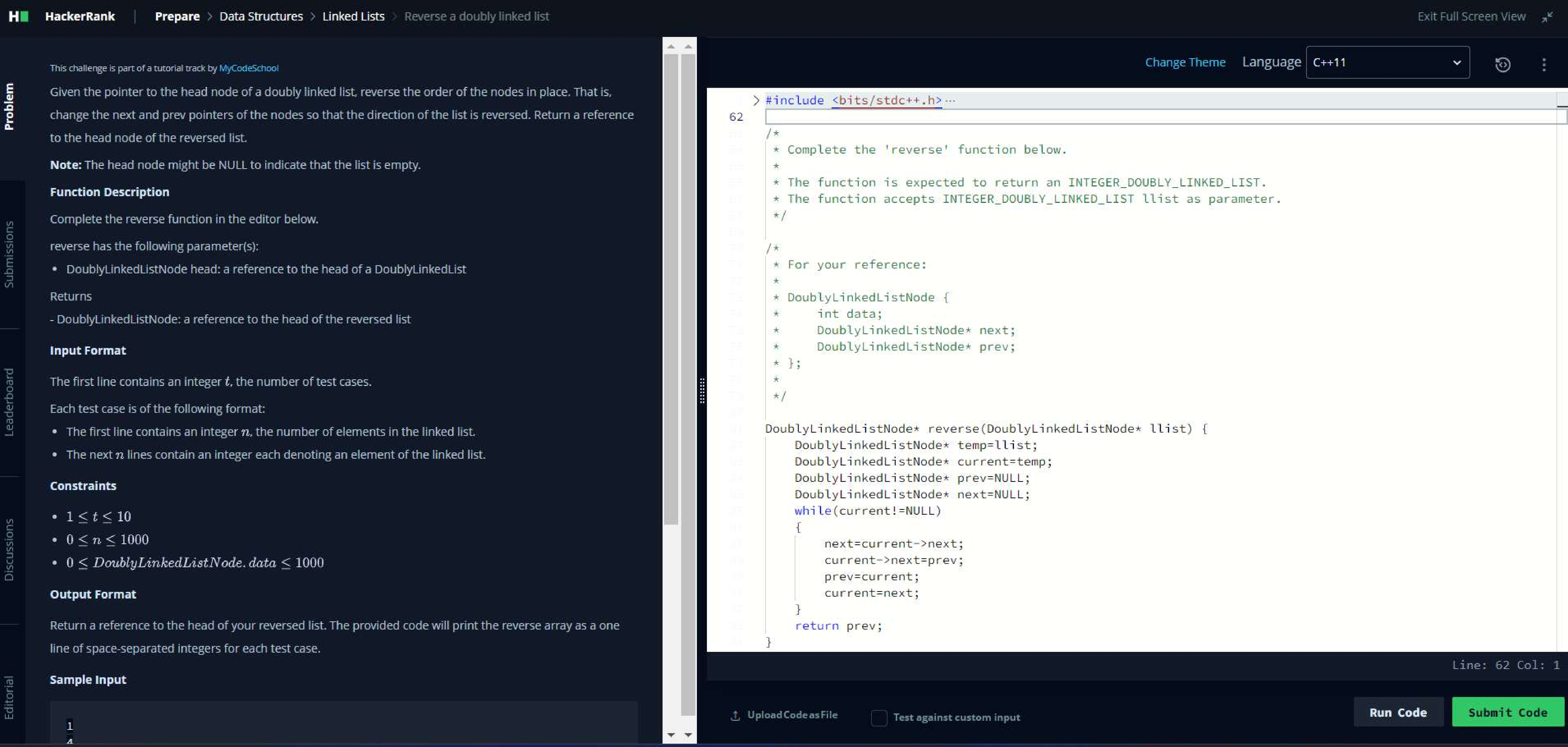


This challenge is part of a tutorial track by MyCodeSchool

Given the pointer to the head node of a doubly linked list, reverse the order of the nodes in place. That is, change the next and prev pointers of the nodes so that the direction of the list is reversed. Return a reference to the head node of the reversed list.

Note: The head node might be NULL to indicate that the list is empty.

Function Description

Complete the reverse function in the editor below.

reverse has the following parameter(s):

- DoublyLinkedListNode head: a reference to the head of a DoublyLinkedList

Returns

- DoublyLinkedListNode: a reference to the head of the reversed list

Input Format

The first line contains an integer t, the number of test cases.

Each test case is of the following format:

- The first line contains an integer n, the number of elements in the linked list.
- The next n lines contain an integer each denoting an element of the linked list.

Constraints

- $1 \leq t \leq 10$
- $0 \leq n \leq 1000$
- $0 \leq DoublyLinkedListNode.data \leq 1000$

Output Format

Return a reference to the head of your reversed list. The provided code will print the reverse array as a one line of space-separated integers for each test case.

Sample Input

```
1
4
```

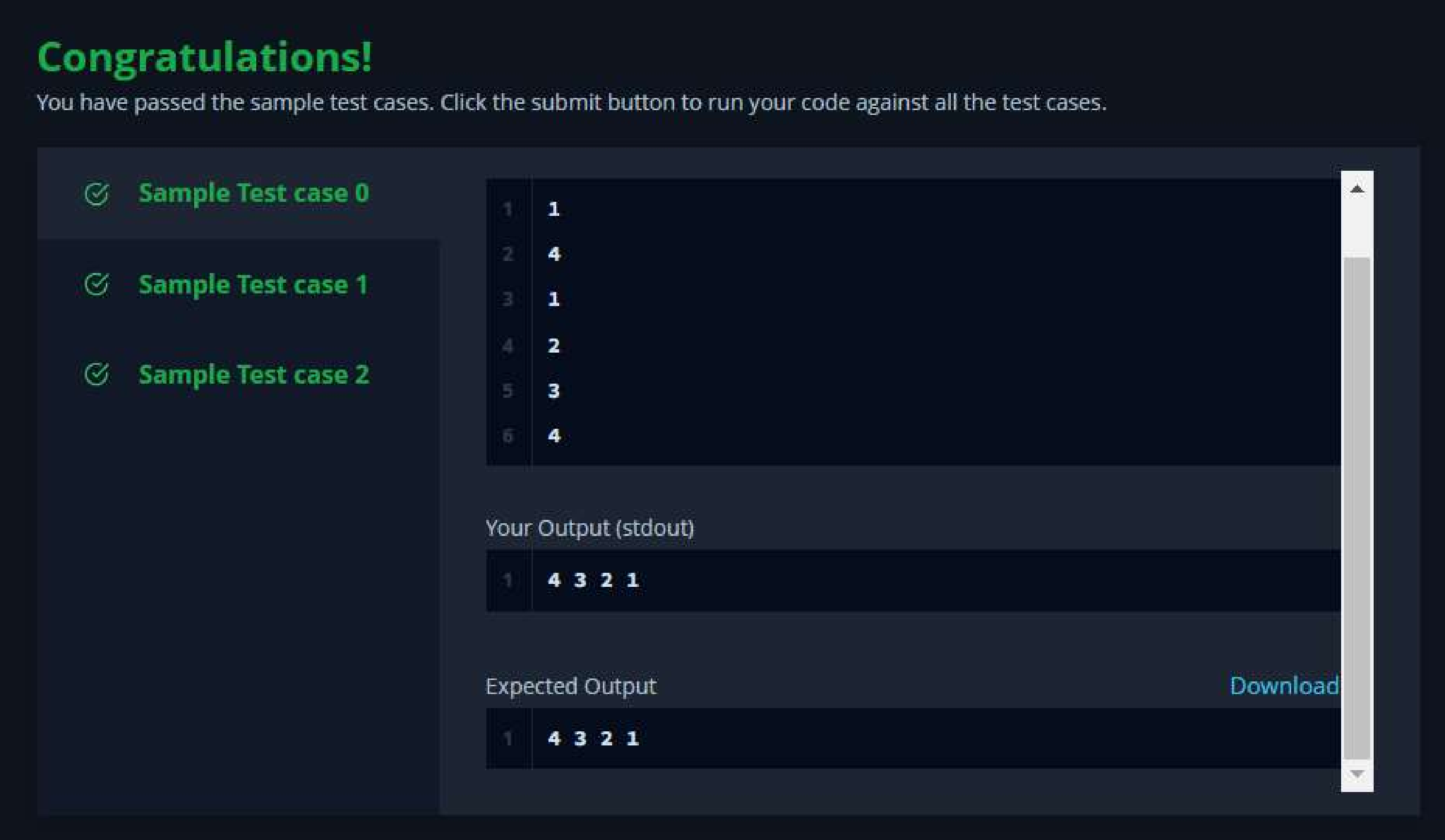
```
#include <bits/stdc++.h> ...

/*
 * Complete the 'reverse' function below.
 *
 * The function is expected to return an INTEGER_DOUBLY_LINKED_LIST.
 * The function accepts INTEGER_DOUBLY_LINKED_LIST llist as parameter.
 */

/*
 * For your reference:
 *
 * DoublyLinkedListNode {
 *     int data;
 *     DoublyLinkedListNode* next;
 *     DoublyLinkedListNode* prev;
 * };
 *
 */

DoublyLinkedListNode* reverse(DoublyLinkedListNode* llist) {
    DoublyLinkedListNode* temp=llist;
    DoublyLinkedListNode* current=temp;
    DoublyLinkedListNode* prev=NULL;
    DoublyLinkedListNode* next=NULL;
    while(current!=NULL)
    {
        next=current->next;
        current->next=prev;
        prev=current;
        current=next;
    }
    return prev;
}
```

Line: 62 Col: 1



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Sample Test case 2

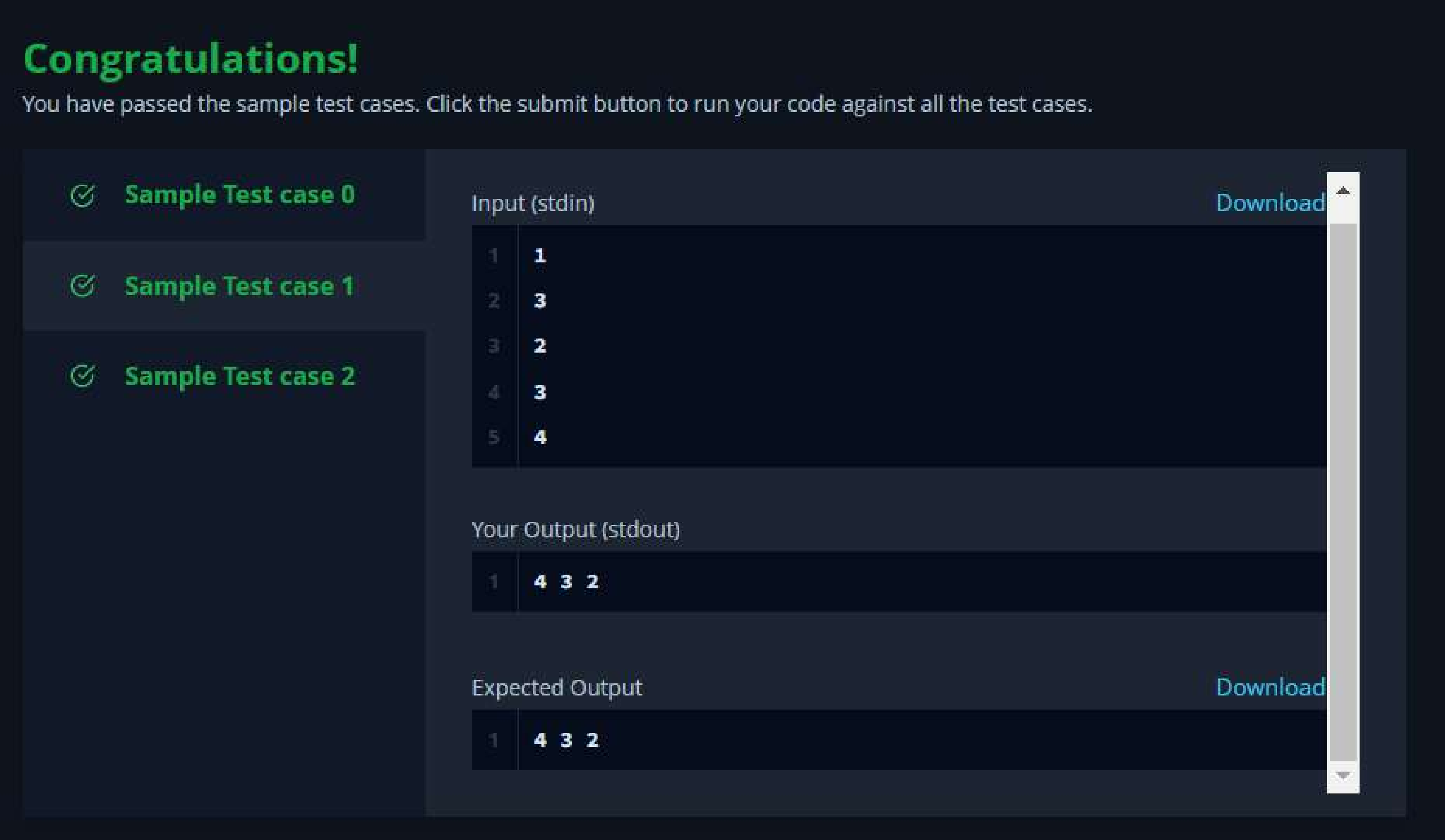
```
1
4
1
2
3
4
```

Your Output (stdout)

```
4 3 2 1
```

Expected Output

```
4 3 2 1
```



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Input (stdin)

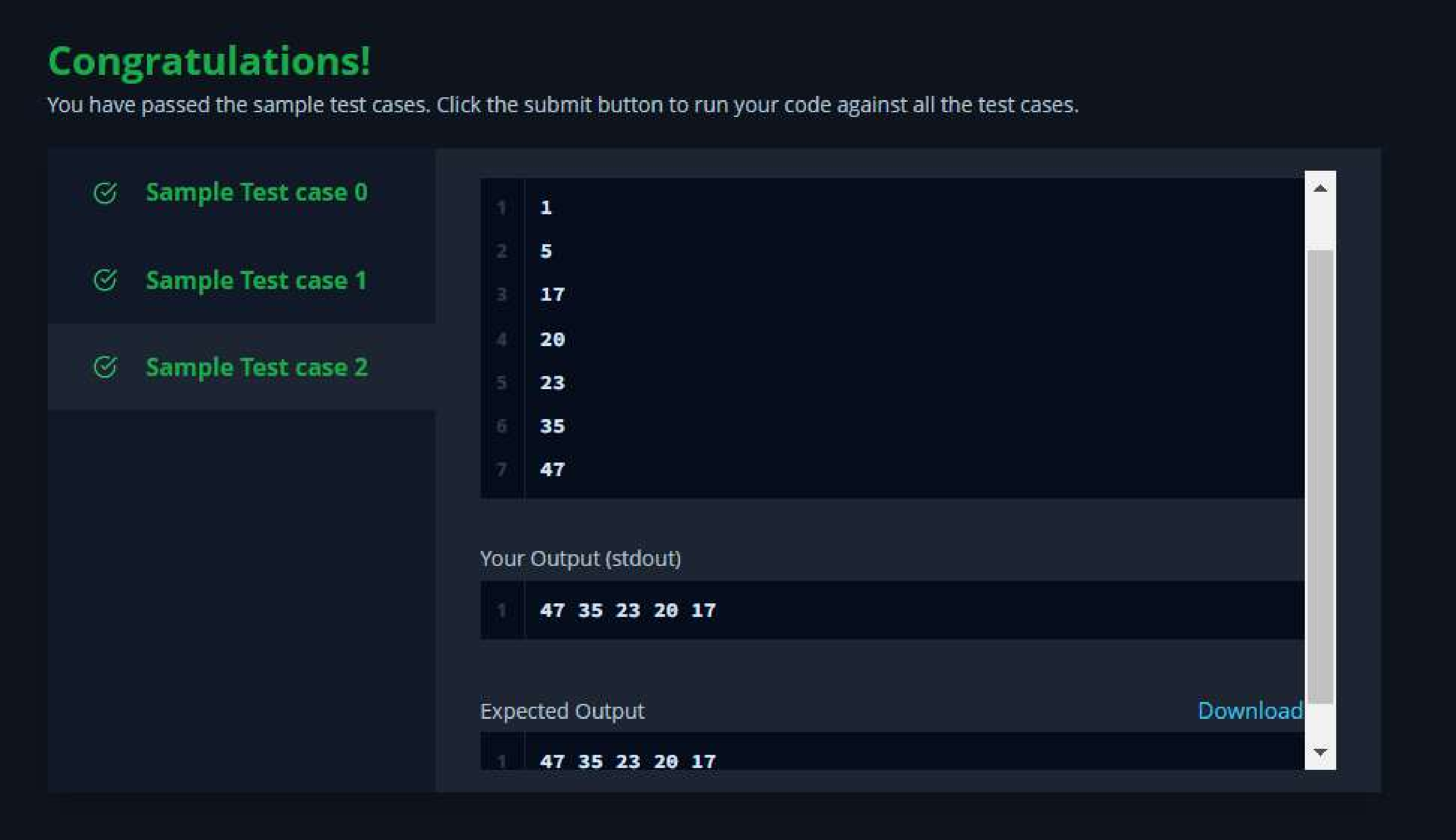
```
1
3
2
3
4
```

Your Output (stdout)

```
4 3 2
```

Expected Output

```
4 3 2
```



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Sample Test case 2

```
1
5
17
20
23
35
47
```

Your Output (stdout)

```
47 35 23 20 17
```

Expected Output

Download

```
47 35 23 20 17
```