

# Reverse a doubly linked list

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

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

## 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
1
2
```

3  
4

### Sample Output

4 3 2 1

### Explanation

The initial doubly linked list is: **1** ↔ **2** ↔ **3** ↔ **4** → *NULL*

The reversed doubly linked list is: **4** ↔ **3** ↔ **2** ↔ **1** → *NULL*