This challenge is part of a MyCodeSchool tutorial track and is accompanied by a video lesson.

This exercise focuses on traversing a linked list. You are given a pointer to the ***head*** node of a linked list. The task is to print the ***data*** of each node, one per line. If the head pointer is ***null***, indicating the list is empty, nothing should be printed.

Function Description

Complete the ***printLinkedList*** function with the following parameter(s):

- ***SinglyLinkedListNode head***: a reference to the head of the list

Print

- For each node, print its ***data*** value on a new line (console.log in Javascript).

Input Format

The first line of input contains ***n***, the number of elements in the linked list.

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

Input (stdin) **Download**

```
1 2
2 16
3 13
```

Your Output (stdout)

```
1 16
2 13
```

Expected Output **Download**

```
1 16
2 13
```

This challenge is part of a tutorial track by MyCodeSchool and is accompanied by a video lesson.

You are given the pointer to the head node of a linked list and an integer to add to the list. Create a new node with the given integer. Insert this node at the tail of the linked list and return the head node of the linked list formed after inserting this new node. The given head pointer may be null, meaning that the initial list is empty.

Function Description

Complete the *insertNodeAtTail* function with the following parameters:

- *SinglyLinkedListNode pointer head*: a reference to the head of a list
- *int data*: the data value for the node to insert

Returns

- *SinglyLinkedListNode pointer*: reference to the head of the modified linked list

Input Format

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

```
5
141
302
164
530
474
```

Your Output (stdout)

```
141
302
164
530
474
```

This challenge is part of a tutorial track by MyCodeSchool and is accompanied by a video lesson.

Given a pointer to the head of a linked list, insert a new node before the head. The $next$ value in the new node should point to $head$ and the $data$ value should be replaced with a given value. Return a reference to the new head of the list. The head pointer given may be null meaning that the initial list is empty.

Function Description

Complete the function $insertNodeAtHead$ with the following parameter(s):

- $SinglyLinkedListNode\ llist$: a reference to the head of a list
- $data$: the value to insert in the $data$ field of the new node

Input Format

The first line contains an integer n, the number of elements to be inserted at the head of the list.
The next n lines contain an integer each, the elements to be inserted, one per function call.

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

```
1  5
2  383
3  484
4  392
5  975
6  321
```

Your Output (stdout)

```
1  321
2  975
3  392
4  484
5  383
```

This challenge is part of a tutorial track by MyCodeSchool and is accompanied by a video lesson.

Delete the node at a given position in a linked list and return a reference to the head node. The head is at position 0. The list may be empty after you delete the node. In that case, return a null value.

Example

$llist = 0 \rightarrow 1 \rightarrow 2 \rightarrow 3$

$position = 2$

After removing the node at position 2, $llist' = 0 \rightarrow 1 \rightarrow 3$.

Function Description

Complete the deleteNode function in the editor below.

deleteNode has the following parameters:

- SinglyLinkedListNode pointer llist: a reference to the head node in the list
- int position: the position of the node to remove

Returns

- SinglyLinkedListNode pointer: a reference to the head of the modified

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

Input (stdin) Download

```
8
20
6
2
19
7
4
15
9
3
```

Your Output (stdout)

This challenge is part of a tutorial track by MyCodeSchool and is accompanied by a video lesson.

Given a pointer to the head of a singly-linked list, print each $data$ value from the reversed list. If the given list is empty, do not print anything.

Example

$head*$ refers to the linked list with $data$ values $1 \rightarrow 2 \rightarrow 3 \rightarrow NULL$

Print the following:

3

2

1

Function Description

Complete the reversePrint function in the editor below.

reversePrint has the following parameters:

- SinglyLinkedListNode pointer head: a reference to the head of the list

Prints

The $data$ values of each node in the reversed list.

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

Input (stdin) Download

```
3
5
16
12
4
2
5
3
7
3
9
5
```