Reverse a linked list iterative

Given a pointer to the head node of a linked list, the task is to reverse the linked list. We need to reverse the list by changing the links between nodes.

Examples:

> ***Input***: *Head of following linked list*
> *1->2->3->4->NULL*
> ***Output***: *Linked list should be changed to,*
> *4->3->2->1->NULL*
>
> ***Input***: *Head of following linked list*
> *1->2->3->4->5->NULL*
> ***Output***: *Linked list should be changed to,*
> *5->4->3->2->1->NULL*

Naive Approach:

C++ Java

```
 import java.io.*;
import java.lang.*;
import java.util.*;

class Node {
    int data;
    Node next;
    Node(int x)
    {
        data = x;
        next = null;
    }
}
```

Track Progress
72 of **132** Complete. (55%)

Dash

All

```
    {
        printlist(head);
        head = revList(head);
        printlist(head);
    }

    static Node revList(Node head)
    {
        ArrayList<Integer> arr = new ArrayList<Integer>();
        for (Node curr = head; curr != null;
             curr = curr.next) {
            arr.add(curr.data);
        }
        for (Node curr = head; curr != null;
             curr = curr.next) {
            curr.data = arr.remove(arr.size() - 1);
        }
        return head;
    }

    public static void printlist(Node head)
    {
        Node curr = head;
        while (curr != null) {
            System.out.print(curr.data + " ");
            curr = curr.next;
        }
        System.out.println();
    }
}
```

Articles

Videos

Problems

Quiz

Contest

Output:

```
10 20 30
30 20 10
```

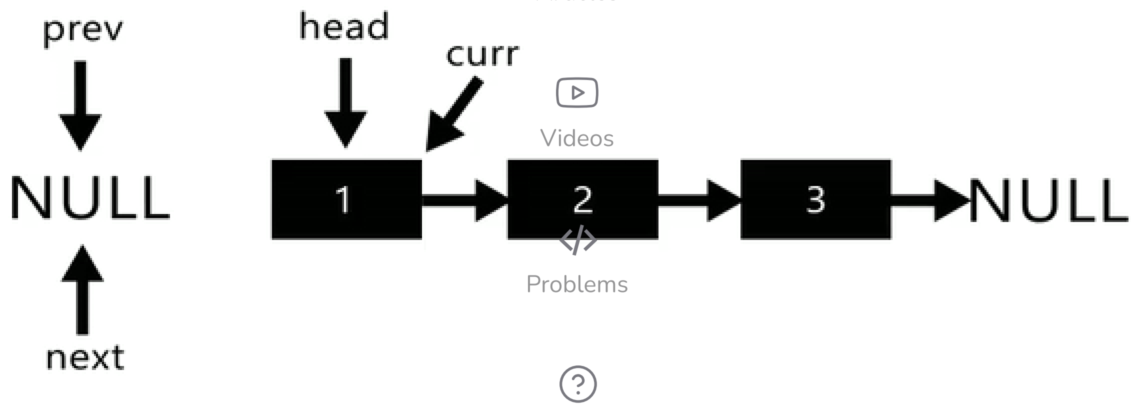
Menu

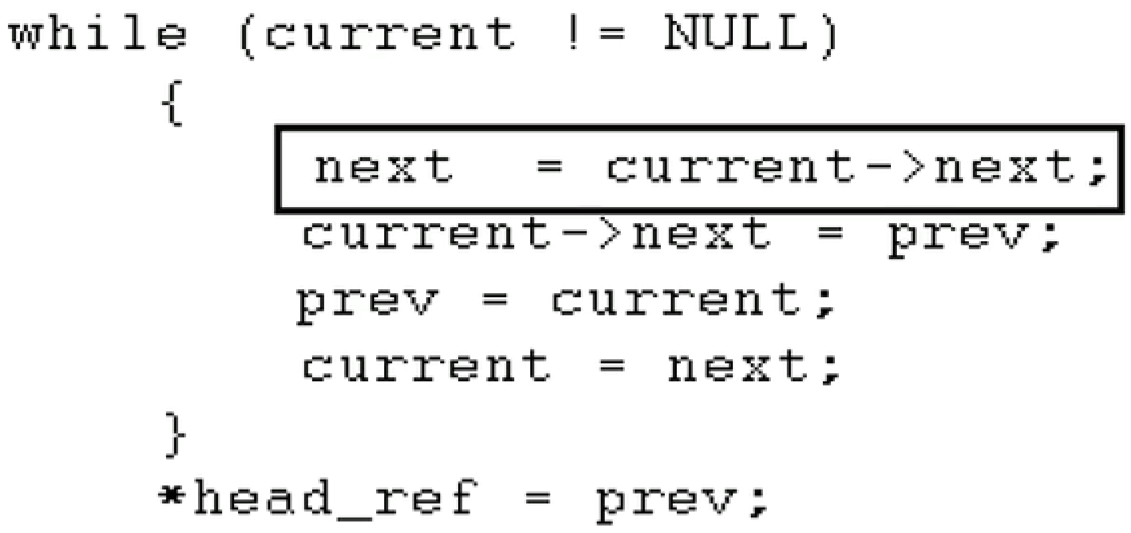
Track Progress

72 of **132** Complete. (55%)

*The idea is to use three pointers **curr**, **prev**, and **next** to keep track of*

Illustration:





```
while (current != NULL)
     {
          next  = current->next;
          current->next = prev;
          prev = current;
          current = next;
     }
     *head_ref = prev;
```

Follow the steps below to solve the problem:

- Initialize three pointers **prev** as NULL, **curr** as **head**, and **next** as NULL.
- Iterate through the linked list. In a loop, do the following:
    - Before changing the **next** of **curr**, store the **next** node
        - next = curr -> next
    - Now update the **next** pointer of **curr** to the **prev**
        - curr -> next = prev

Below is the implementation of the above approach:

```
// Java program for reversing the linked list

class LinkedList {

    static Node head;

    static class Node {

        int data;
        Node next;

        Node(int d)
        {
            data = d;
            next = null;
        }
    }

    /* Function to reverse the linked list */
    Node reverse(Node node)
    {
        Node prev = null;
        Node current = node;
        Node next = null;
        while (current != null) {
            next = current.next;
            current.next = prev;
            prev = current;
            current = next;
        }
        node = prev;
        return node;
    }

    // prints content of double linked list
    void printList(Node node)
```

Track Progress
72 of **132** Complete. (55%)

```
            node = node.next;
```

```
    // Driver Code
    public static void main(String[] args)
    {
        LinkedList list = new LinkedList();
        list.head = new Node(85);
        list.head.next = new Node(15);
        list.head.next.next = new Node(4);
        list.head.next.next.next = new Node(20);

        System.out.println("Given linked list");
```

```
        System.out.println("");
        System.out.println("Reversed linked list ");
        list.printList(head);
    }
}
```

Output

```
Given linked list
85 15 4 20
Reversed linked list
20 4 15 85
```

Time Complexity: O(N), Traversing over the linked list of size N.
Auxiliary Space: O(1)

Dash
Report An Issue

All

Articles

Videos

Problems

Quiz

Contest

Menu

Track Progress

72 of **132** Complete. (55%)