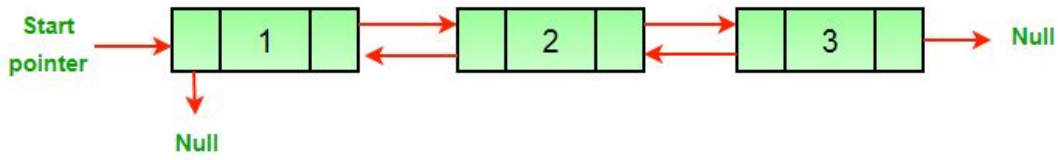
Reverse a Doubly Linked List

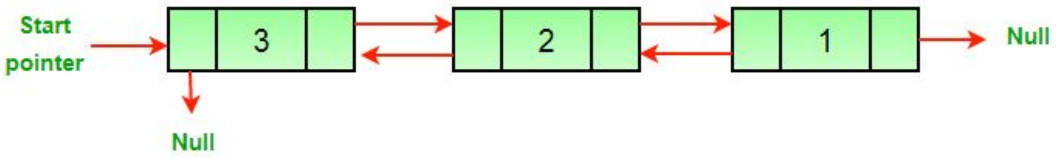
Given a Doubly Linked List, the task is to reverse the given Doubly Linked List.

Example:

Input:



Output:



Follow the given steps to solve the problem using the above approach:

- Traverse the linked list using a pointer
- Swap the prev and next pointers for all nodes
- At last, change the head pointer of the doubly linked list

Below is the implementation of the above approach:

C++ | Java

```
// Java program to reverse a doubly linked list

class LinkedList {

    static Node head;
```

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

/* Function to reverse a Doubly Linked List */
void reverse()
{
    Node temp = null;
    Node current = head;

    /* swap next and prev for all nodes of
    doubly linked list */
    while (current != null) {
        temp = current.prev;
        current.prev = current.next;
        current.next = temp;
        current = current.prev;
    }

    /* Before changing head, check for the cases like
    empty list and list with only one node */
    if (temp != null) {
        head = temp.prev;
    }
}

/* UTILITY FUNCTIONS */
/* Function to insert a node at the beginning of the
* Doubly Linked List */
void push(int new_data)
{
    /* allocate node */
    Node new_node = new Node(new_data);
```

```
        new_node.prev = null;

        /* change prev of head node to new node */
        if (head != null) {
            head.prev = new_node;
        }

        /* move the head to point to the new node */
        head = new_node;
    }

    /* Function to print nodes in a given doubly linked list
    This function is same as printList() of singly linked
    list */
    void printList(Node node)
    {
        while (node != null) {
            System.out.print(node.data + " ");
            node = node.next;
        }
    }

    // Driver's code
    public static void main(String[] args)
    {
        LinkedList list = new LinkedList();

        /* Let us create a sorted linked list to test the
        functions Created linked list will be 10->8->4->2
    */
        list.push(2);
        list.push(4);
        list.push(8);
        list.push(10);

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

```
    }
}
```

Output

```
Original Linked list
10 8 4 2
Reversed Linked list
2 4 8 10
```

Time Complexity: O(N), where N denotes the number of nodes in the doubly linked list.
Auxiliary Space: O(1)