Karthick _ M _ AI&DS _ DSA-Practice

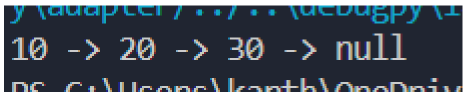
Linked List

Singly Linked List creation:

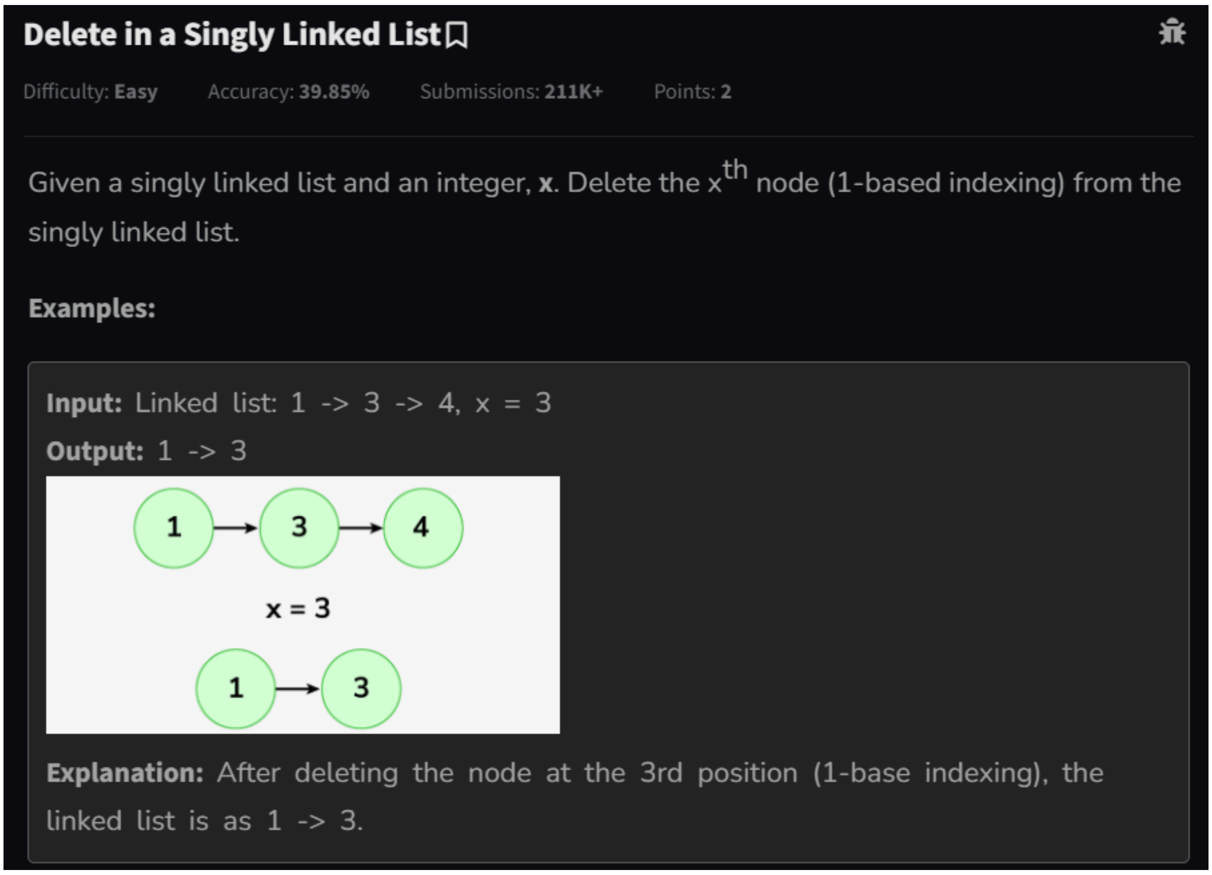
Program:

```
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
        this.next = null;
    }
}

class SinglyLinkedList {
    private Node head;

    public void append(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node current = head;
        while (current.next != null) {
            current = current.next;
        }
        current.next = newNode;
    }

    public void display() {
        if (head == null) {
            System.out.println("The list is empty.");
            return;
        }
        Node current = head;
        while (current != null) {
            System.out.print(current.data + " -> ");
            current = current.next;
        }
        System.out.println("null");
    }
}

public class Main {
    public static void main(String[] args) {
        SinglyLinkedList sll = new SinglyLinkedList();
        sll.append(10);
        sll.append(20);
        sll.append(30);
        sll.display();
    }
}
```

Output:



```
10 -> 20 -> 30 -> null
```



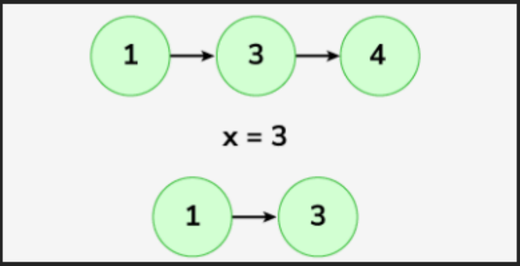
Delete in a Singly Linked List

Difficulty: **Easy** Accuracy: **39.85%** Submissions: **211K+** Points: **2**

Given a singly linked list and an integer, **x**. Delete the xth node (1-based indexing) from the singly linked list.

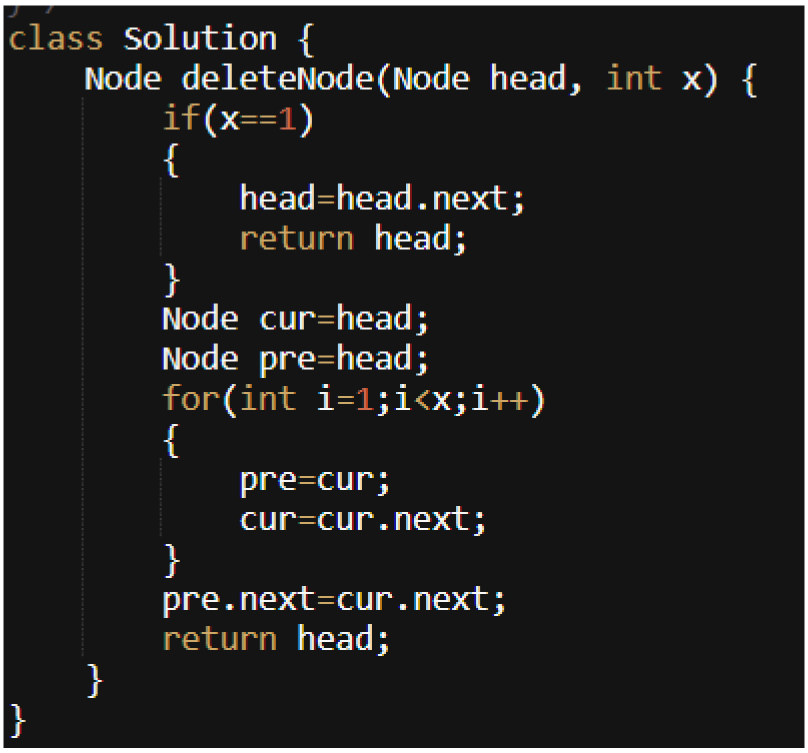
Examples:

Input: Linked list: 1 -> 3 -> 4, x = 3
Output: 1 -> 3



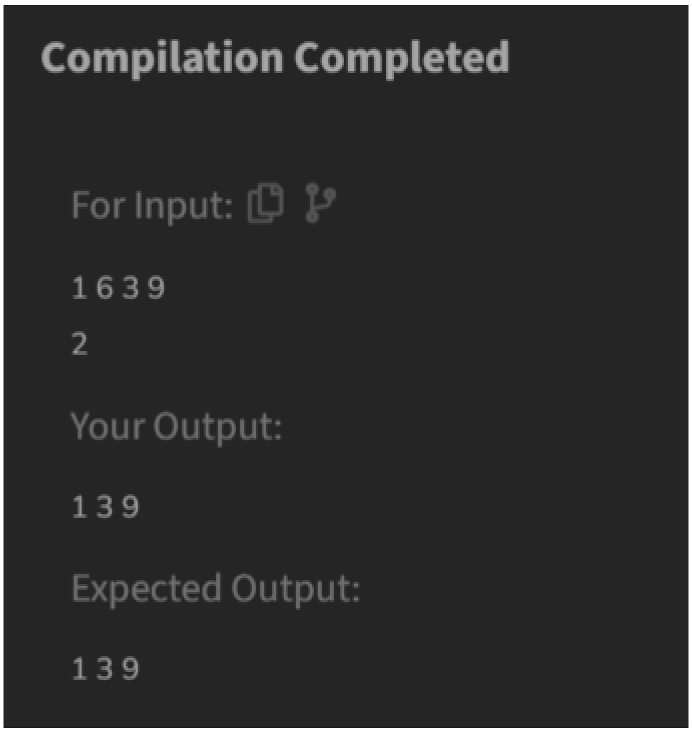
Explanation: After deleting the node at the 3rd position (1-base indexing), the linked list is as 1 -> 3.

Program:



```java
class Solution {
    Node deleteNode(Node head, int x) {
        if(x==1)
        {
            head=head.next;
            return head;
        }
        Node cur=head;
        Node pre=head;
        for(int i=1;i<x;i++)
        {
            pre=cur;
            cur=cur.next;
        }
        pre.next=cur.next;
        return head;
    }
}
```

Output:



Compilation Completed

For Input:

```
1 6 3 9
2
```

Your Output:

```
1 3 9
```

Expected Output:

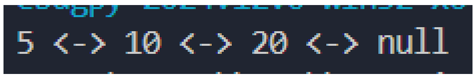
```
1 3 9
```

Insertion in Doubly Linked List :

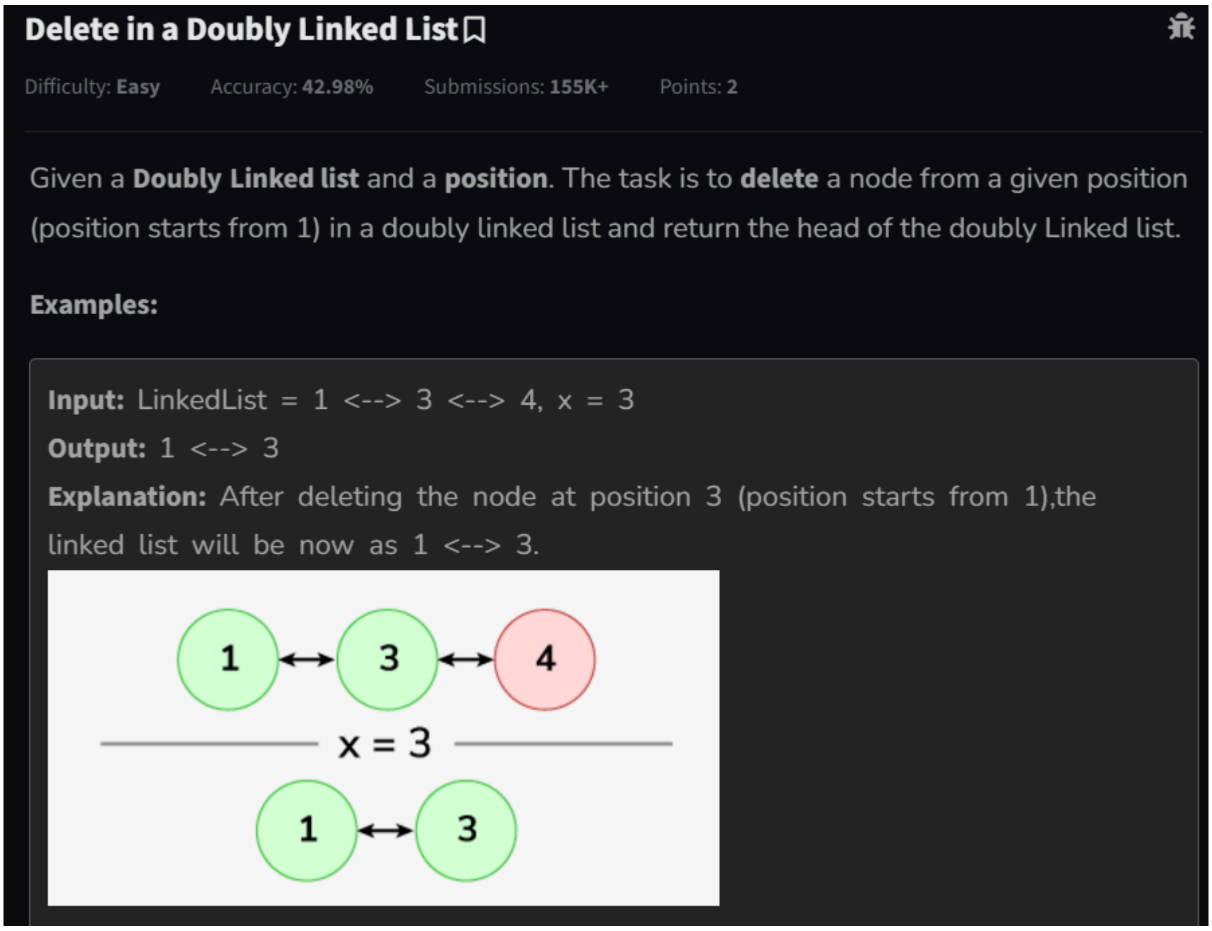
Program:

```
class Node {
    int data;
    Node prev, next;

    Node(int data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}

class DoublyLinkedList {
    private Node head;

    public void insertAtEnd(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node current = head;
        while (current.next != null) {
            current = current.next;
        }
        current.next = newNode;
        newNode.prev = current;
    }

    public void insertAtBeginning(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        newNode.next = head;
        head.prev = newNode;
        head = newNode;
    }

    public void display() {
        Node current = head;
        while (current != null) {
            System.out.print(current.data + " <-> ");
            current = current.next;
        }
        System.out.println("null");
    }
}

public class Main {
    public static void main(String[] args) {
        DoublyLinkedList dll = new DoublyLinkedList();
        dll.insertAtEnd(10);
        dll.insertAtEnd(20);
        dll.insertAtBeginning(5);
        dll.display();
    }
}
```

Output:



```
5 <-> 10 <-> 20 <-> null
```



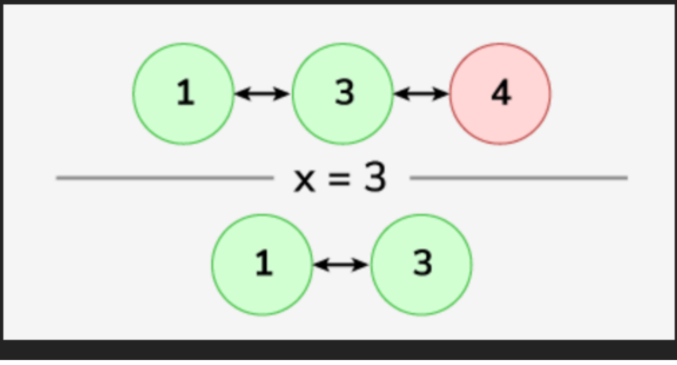
Delete in a Doubly Linked List

Difficulty: **Easy** Accuracy: **42.98%** Submissions: **155K+** Points: **2**

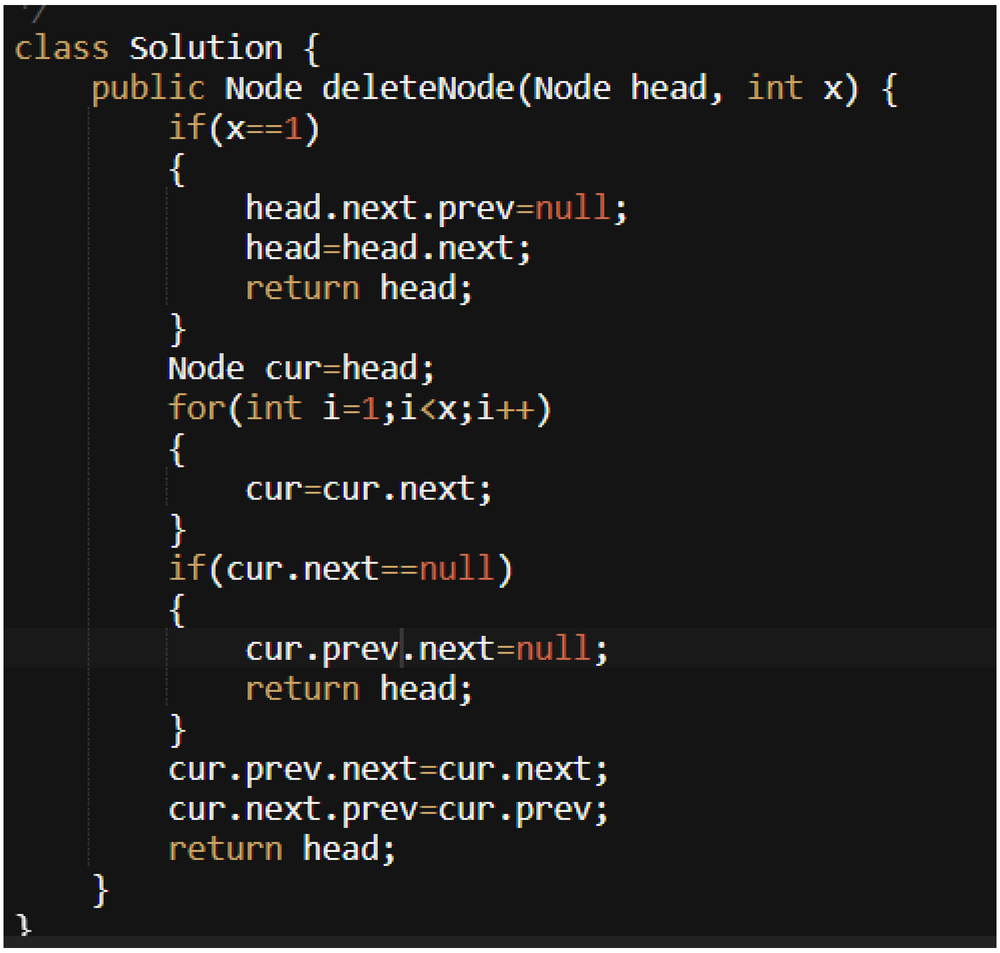
Given a **Doubly Linked list** and a **position**. The task is to **delete** a node from a given position (position starts from 1) in a doubly linked list and return the head of the doubly Linked list.

Examples:

Input: LinkedList = 1 <--> 3 <--> 4, x = 3
Output: 1 <--> 3
Explanation: After deleting the node at position 3 (position starts from 1),the linked list will be now as 1 <--> 3.

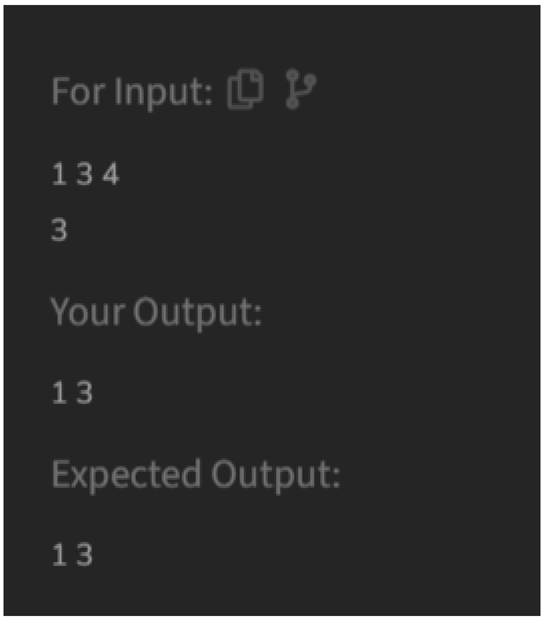


Program:



```
class Solution {
    public Node deleteNode(Node head, int x) {
        if(x==1)
        {
            head.next.prev=null;
            head=head.next;
            return head;
        }
        Node cur=head;
        for(int i=1;i<x;i++)
        {
            cur=cur.next;
        }
        if(cur.next==null)
        {
            cur.prev.next=null;
            return head;
        }
        cur.prev.next=cur.next;
        cur.next.prev=cur.prev;
        return head;
    }
}
```

Output:



```
For Input:

1 3 4
3

Your Output:

1 3

Expected Output:

1 3
```