DSA Linked List Practice

By Jashwanth SA (22CS066)

1. **Single linked list operations: (Insertion, Deletion, Delete from middle)**

```
class Node {
    int key;
    Node next;

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

}
public class singleLL{

    static Node insert(Node root,int key){
        Node new_node=new Node(key);
        if(root==null){
            return new_node;
        }else{
            Node last=root;
            while(last.next!=null){last=last.next;}
            last.next=new_node;
        }
        return root;
    }
    static void deleteFromMiddle(Node root){
        if(root==null || root.next==null){return;}
        Node fast=root,slow=root,prev=null;
        while(fast!=null && fast.next!=null){
            fast=fast.next.next;
            prev=slow;
            slow=slow.next;
        }
        prev.next=slow.next;
    }
    static void delete(Node root,int val){
        if(root==null) return;
        Node last=root;
        while (last.next!=null) {
            if(last.next.key==val){
                last.next=last.next.next;
                return;
            }
            last=last.next;
        }
    }
    static void SllPrint(Node root){
        while (root!=null) {
            System.out.print(root.key+" ");
            root=root.next;
        }
```

```
    }
    public static void main(String[] args){
        Node root=null;
        root=insert(root, 50);
        root=insert(root, 70);
        root=insert(root, 30);
        root=insert(root, 20);
        root=insert(root,40);
        delete(root, 30);
        deleteFromMiddle(root);
        SllPrint(root);
    }
}
```

Output: 50 70 40

2. **Double linked list operations: (Insertion, Deletion, Delete from middle)**

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

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

public class doubleLL {
    static Node insert(Node root, int val){
        Node new_node = new Node(val);
        if (root == null) {
            return new_node;
        }else{
            Node last = root;
            while(last.next!=null){last=last.next;}
            last.next = new_node;
        }
        return root;
    }
    static void delete(Node root,int val){
        if(root==null) return;
        Node last=root;
        while(last.next!=null){
            if(last.next.data==val){last.next=last.next.next;return;}
            last=last.next;
        }
    }
    static void deleteFromMiddle(Node root){
        if(root==null || root.next==null) return;
        Node slow = root, fast = root, prev=null;
        while (fast != null && fast.next != null) {
            fast=fast.next.next;
            prev=slow;
```

```
            slow=slow.next;
        }
        if (prev != null) {
            prev.next = slow.next;
            if (slow.next != null) {
                slow.next.prev = prev;
            }
        }
    }
    static void Dllprint(Node root){
        while (root!=null) {
            System.out.print(root.data+" ");
            root=root.next;
        }
    }
    public static void main(String[] args){
        Node root=null;
        root=insert(root, 50);
        root=insert(root, 70);
        root=insert(root, 30);
        root=insert(root, 20);
        root=insert(root,40);
        delete(root, 30);
        deleteFromMiddle(root);
        Dllprint(root);
    }
}
```

Output: 50 70 40