GATE TECHNICAL TRAINING – DSA CODING PRACTICE PROBLEMS 2026

DATE: 29-11-2024 **NAME: JASHVARTHINI R – CSBS**

1. SINGLY LINKED LIST

```
class SLL {
    public static Node fun(Node root) {
        if (root == null || root.next == null) {
            return root;
        }

        // Create 6 and append it at the end
        Node n = new Node(6);
        Node temp = root;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = n;

        // Delete node with value 5
        temp = root;
        while (temp != null && temp.next != null) {
            if (temp.next.data == 5) {
                temp.next = temp.next.next;
            } else {
                temp = temp.next;
            }
        }

        Node slow = root;
        Node fast = root;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        int mid = slow.data;
        System.out.println("Middle element: " + mid);

        return root;
    }

    public static void main(String[] args) {
        // Initialize the linked list
        Node root = new Node(1);
        root.next = new Node(2);
        root.next.next = new Node(3);
```

```
        root.next.next.next = new Node(4);
        root.next.next.next.next = new Node(5);

        // Modify the list
        root = fun(root);

        // Print updated linked list
        Node temp = root;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
    }
}

class Node {
    int data;
    Node next;

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

OUTPUT:

```
Middle element: 3
1 2 3 4 6
```

2. DOUBLY LINKED LIST

```
class DLL {
    public static Node fun(Node root) {
        if (root == null || root.next == null) return root;
        Node n = new Node(6);
        Node temp = root;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = n;
        n.prev = temp;

    //delete 5
    temp = root;
```

```
    while(temp!= null && temp.next != null){
        if(temp.next.data == 5){
            temp.next = temp.next.next;
        }
        temp = temp.next;
    }

    //mid
    Node slow = root;
    Node fast = root;

    while(fast != null && fast.next != null){
        Node a = slow;
        Node b = fast;
        slow = slow.next;
        fast = fast.next.next;
        slow.prev = a;
        fast.prev = b;
    }
    System.out.println("Middle element: " + slow.data);
    return root;

}
    public static void main(String[] args) {
        Node a = new Node(1);
        Node b = new Node(2);
        Node c = new Node(3);
        Node d = new Node(4);
        Node e = new Node(5);

        Node root = a;
        a.next = b;
        a.prev = null;

        b.next = c;
        b.prev = a;

        c.next = d;
        c.prev = b;

        d.next = e;
        d.prev = c;

        e.next = null;
        e.prev = d;

        root = fun(root);
    }
```

```
}

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

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

OUTPUT:

```
Middle element: 3
```