1)Create singly linked list

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

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

class SinglyLinkedList {
    Node head;

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

    void insertAtBeginning(int data) {
        Node newNode = new Node(data);
        newNode.next = head;
        head = newNode;
    }

    void deleteByKey(int key) {
```

```
        if (head == null) return;
        if (head.data == key) {
            head = head.next;
            return;
        }
        Node temp = head;
        while (temp.next != null && temp.next.data != key) {
            temp = temp.next;
        }
        if (temp.next != null) {
            temp.next = temp.next.next;
        }
    }

    void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}

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

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\5>javac Main.java

C:\Users\ASUS\OneDrive\Desktop\5>java Main
5 10 20
5 10

C:\Users\ASUS\OneDrive\Desktop\5>
```

Time Complexity:

Insert at the end: O(n)

Insert at the beginning: O(1)

Delete by key: O(n)

Display: O(n)

2)Creation of doubly linked list

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

    DoublyNode(int data) {
        this.data = data;
        this.prev = null;
        this.next = null;
```

```
    }
}

class DoublyLinkedList {
    DoublyNode head;

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

    void insertAtBeginning(int data) {
        DoublyNode newNode = new DoublyNode(data);
        if (head != null) {
            head.prev = newNode;
        }
        newNode.next = head;
        head = newNode;
    }

    void deleteByKey(int key) {
        if (head == null) return;
        if (head.data == key) {
            head = head.next;
            if (head != null) head.prev = null;
            return;
```

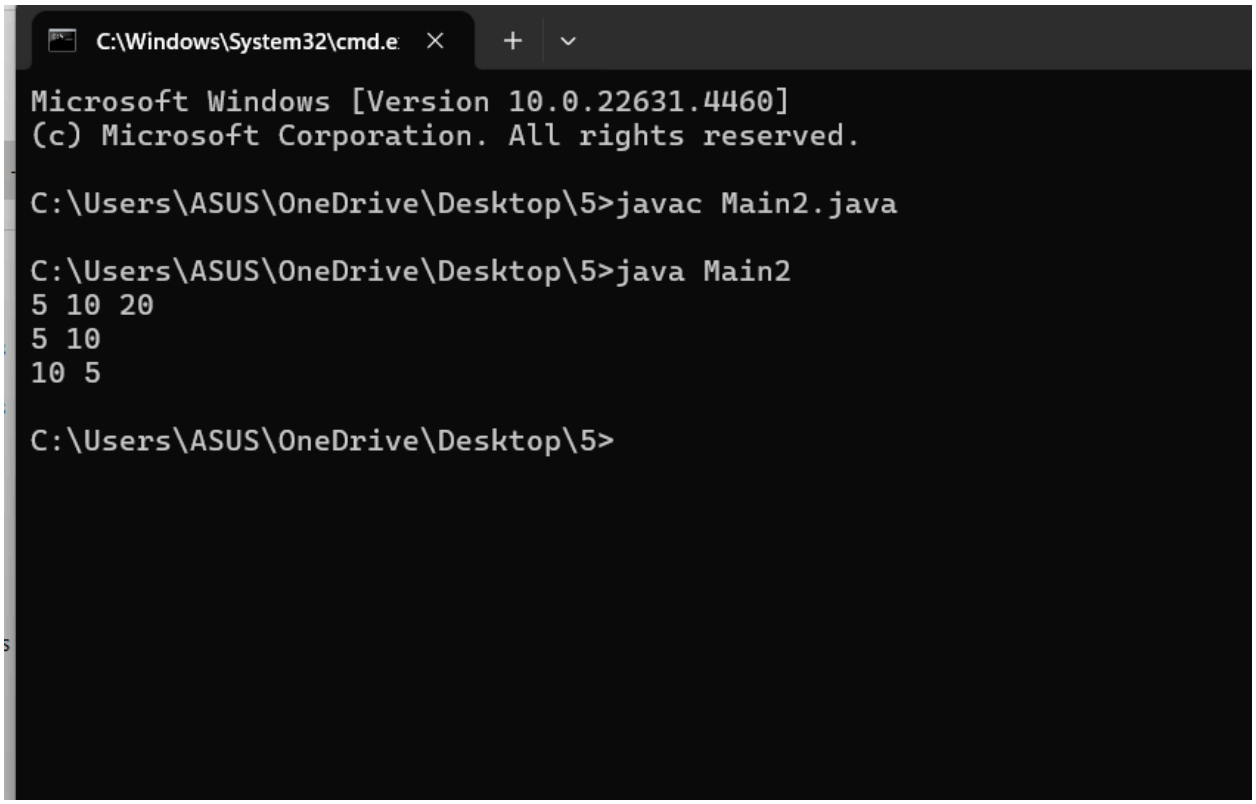
```
        }
        DoublyNode temp = head;
        while (temp != null && temp.data != key) {
            temp = temp.next;
        }
        if (temp != null) {
            if (temp.next != null) temp.next.prev = temp.prev;
            if (temp.prev != null) temp.prev.next = temp.next;
        }
    }

    void displayForward() {
        DoublyNode temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }

    void displayBackward() {
        if (head == null) return;
        DoublyNode temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.prev;
        }
        System.out.println();
    }
}

public class Main {
```

```
    public static void main(String[] args) {
        DoublyLinkedList list = new DoublyLinkedList();
        list.insertAtEnd(10);
        list.insertAtEnd(20);
        list.insertAtBeginning(5);
        list.displayForward();
        list.deleteByKey(20);
        list.displayForward();
        list.displayBackward();
    }
}
```



Time Complexity:

Insert at the end: O(n)

Insert at the beginning: O(1)

Delete by key: O(n)

Display forward/backward: O(n)

3)insertion,deletion,deleting from middle

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

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

class DoublyLinkedList {
    DoublyNode head;

    void insertAtBeginning(int data) {
        DoublyNode newNode = new DoublyNode(data);
        if (head != null) {
            head.prev = newNode;
        }
        newNode.next = head;
        head = newNode;
    }

    void insertAtEnd(int data) {
        DoublyNode newNode = new DoublyNode(data);
        if (head == null) {
            head = newNode;
            return;
        }
```

```
    DoublyNode temp = head;
    while (temp.next != null) {
        temp = temp.next;
    }
    temp.next = newNode;
    newNode.prev = temp;
}

void deleteFromBeginning() {
    if (head == null) return;
    head = head.next;
    if (head != null) {
        head.prev = null;
    }
}

void deleteFromEnd() {
    if (head == null) return;
    if (head.next == null) {
        head = null;
        return;
    }
    DoublyNode temp = head;
    while (temp.next != null) {
        temp = temp.next;
    }
    temp.prev.next = null;
}

void deleteFromMiddle(int key) {
    if (head == null) return;
    if (head.data == key) {
        deleteFromBeginning();
        return;
    }
```

```
        DoublyNode temp = head;
        while (temp != null && temp.data != key) {
            temp = temp.next;
        }
        if (temp != null) {
            if (temp.next != null) temp.next.prev = temp.prev;
            if (temp.prev != null) temp.prev.next = temp.next;
        }
    }

    void displayForward() {
        DoublyNode temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}

public class Main {
    public static void main(String[] args) {
        DoublyLinkedList list = new DoublyLinkedList();
        list.insertAtBeginning(10);
        list.insertAtEnd(20);
        list.insertAtEnd(30);
        list.insertAtBeginning(5);
        list.displayForward();

        list.deleteFromBeginning();
        list.displayForward();

        list.deleteFromEnd();
        list.displayForward();
```

```
        list.deleteFromMiddle(20);
        list.displayForward();
    }
}
```

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\5>javac Main3.java

C:\Users\ASUS\OneDrive\Desktop\5>java Main3
5 10 20 30
10 20 30
10 20
10

C:\Users\ASUS\OneDrive\Desktop\5>
```

Time Complexity

Insert at beginning: O(1)

Insert at end: O(n)

Delete from beginning: O(1)

Delete from end: O(n)

Delete from middle: O(n)

Display forward: O(n).