Flowchart

start

Node head, tail, data, prev, next

Data = d, head = null, tail = null, newNode = new node(data)

addinFront

addNode

head ==nul

yes

Head = tail = newNode, head.prev =null, tail.next = null

no

Tail.next =newNode, newNode.prev= tail, tail = newNode, tail.next = null

Node newNode = new Node(data), newNode.next = head, newNode.prev = null

Head != null

head.prev = newNode, head = newNode

addAfter

Prev_Node == null

display("previous node cannot be null"

return

1

2

3

4

① ② ③ ④

Node new_node = new Node(data, new_node.next = prev_node.next, prev_node.next = new_node, new_node.prev = prev_node

new_node. next != nul

new_node.next.prev = new_nod

Node new_node = new Node(data, Node temp = head, new_node.next = null

insertEnd

head == null

new_node.prev = nul, head = new_node, return

while (temp.next != null)

temp = temp.next, temp.next = new_node, new_node.prev = temp

① ②

```
        ( 1 )                                                    ( 2 )

                if(head ==                      return
                   null

                   Node current = head, int
                            pos =n

                for(int i = 1; i <              current = current.next
                        pos; i++

                if(current         yes          head = current.next
                  == head)

                        no

                   else
                if(current ==      yes          tail = tail.prev
                      tail

                        no

         current.prev.next = current.next, current.next.prev =
                    current.prev, current = null

        ( 1 )                                                    ( 2 )
```

( 1 )

( 2 )

Node last = null

while (node != null) → display(node.data + "-->"), System.*out*.println()

last = node, node = node.next

int i = 1, boolean flag = false, Node current = head

if (head == null) — yes → Display("List is empty"

int fnd = 0

return

while (current != null)

( 1 )

( 2 )

```
                                    1                                           2

         if                              Display("the same ", ("Node is
    ((current.dat                        present in the list at the position
    a.equals(val

    current =                            ("Node is not present in the list");
    current.next, i++

printReverse

    Node tail = head_ref

         while                           tail = tail.next
    (tail.next !=
       null)

    while (tail !=                       Display( tail.data + "--> , (
     head_ref)                           tail.data )"),

                                         tail = tail.prev

                        END
```