Rishikesh RR
A776647

1. Inserting a node at the tail of the linked list

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

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

class LinkedList {
  Node head;

  void insertAtTail(int data) {
    if (head == null) head = new Node(data);
    else {
      Node current = head;
      while (current.next != null) current = current.next;
      current.next = new Node(data);
    }
  }

  void printList() {
    for (Node current = head; current != null; current = current.next)
      System.out.print(current.data + " ");
    System.out.println();
  }
}

public class Main {
  public static void main(String[] args) {
    LinkedList list = new LinkedList();
    for (int value : new int[]{141, 302, 164, 530, 474})
      list.insertAtTail(value);
    list.printList();
  }
}
```

2.Inserting a node at the head of the linked list

```
package Linked_list;

class Node {
  int data;
  Node next;

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

class LinkedList {
  Node head;

  void insertAtHead(int data) {
    head = new Node(data) {{ next = head; }};
  }

  void printList() {
    for (Node current = head; current != null; current = current.next)
      System.out.print(current.data + " ");
    System.out.println();
  }
}

public class Deletion {
  public static void main(String[] args) {
    LinkedList list = new LinkedList();
    for (int value : new int[]{383, 484, 392, 975, 321})
      list.insertAtHead(value);
    list.printList();
  }
}
```

3.Insert a node at a specific position

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

   Node(int data) {
```

```
        this.data = data;
    }
}

class LinkedList {
    Node head;

    void insertAtPosition(int data, int position) {
        Node newNode = new Node(data);
        if (position == 0) {
            newNode.next = head;
            head = newNode;
            return;
        }
        Node current = head;
        for (int i = 0; i < position - 1 && current != null; i++) {
            current = current.next;
        }
        if (current == null) throw new IndexOutOfBoundsException("Position out of
bounds");
        newNode.next = current.next;
        current.next = newNode;
    }

    void printList() {
        for (Node current = head; current != null; current = current.next)
            System.out.print(current.data + " ");
        System.out.println();
    }
}

public class Main {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        int[] values = {16, 13, 7};
        for (int i = 0; i < values.length; i++) {
            list.insertAtPosition(values[i], i);
        }
        list.insertAtPosition(1, 2);
        list.printList();
    }
}
```

4.Tree Preorder traversal

```
class Node {
   int data;
   Node left, right;
   Node(int d) { data = d; }
}

class BinaryTree {
   Node root;

   void preOrder(Node node) {
      if (node != null) {
         System.out.print(node.data + " ");
         preOrder(node.left);
         preOrder(node.right);
      }
   }

   void preOrderTraversal() {
      preOrder(root);
   }
}

public class Main {
   public static void main(String[] args) {
      BinaryTree tree = new BinaryTree();
      tree.root = new Node(1);
      tree.root.right = new Node(2);
      tree.root.right.right = new Node(5);
      tree.root.right.right.left = new Node(3);
      tree.root.right.right.right = new Node(6);
      tree.root.right.right.left.right = new Node(4);
      tree.preOrderTraversal();
   }
}
```

5. Reverse the element of a linked list.

```
class Node {
```

```
    int data;
    Node next;

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

class LinkedList {
    Node head;

    void insertAtTail(int data) {
        if (head == null) head = new Node(data);
        else {
            Node current = head;
            while (current.next != null) current = current.next;
            current.next = new Node(data);
        }
    }

    void reverse() {
        Node prev = null, current = head, next;
        while (current != null) {
            next = current.next;
            current.next = prev;
            prev = current;
            current = next;
        }
        head = prev;
```

```
    }

    void printList() {
        for (Node current = head; current != null; current = current.next)
            System.out.print(current.data + " ");
        System.out.println();
    }
}

public class Main {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        for (int value : new int[]{1, 2, 3, 4, 5})
            list.insertAtTail(value);

        System.out.println("Original list:");
        list.printList();
        list.reverse();
        System.out.println("Reversed list:");
        list.printList();
    }
}
```