Student Name: Pulkit
UID: 23BCS11733
Branch: CSE
Section/Group: KRG-3B
Semester: 5th
Date of Performance: 24 JUL
Subject Name: Design and Analysis of Algorithm (23CSH-301)
Experiment No: 2

AIM: Operations on Linked List

1. Insertion in Singly Linked List

Algorithm:

- Create a new node with value x
- If inserting at the beginning (pos == 0):
  - Set new node's next to head
  - Set head to the new node
- Else:
  - Traverse to the node at pos - 1
  - Set new node's next to current.next
  - Set current.next to the new node

Java Code:

```java
class Node {
    int data;
    Node next;
    Node(int d) {
        data = d;
        next = null;
    }
}

public void insertAtPosition(Node head, int x, int pos) {
    Node newNode = new Node(x);
    if (pos == 0) {
        newNode.next = head;
        head = newNode;
        return;
    }
    Node current = head;
    for (int i = 0; current != null && i < pos - 1; i++) {
        current = current.next;
    }
    if (current == null) return;
    newNode.next = current.next;
    current.next = newNode;
}
```

2. Deletion in Singly Linked List

Algorithm:

- If list is empty, return
- If pos == 0:
  - Move head to head.next
  - Delete the original head
- Else:
  - Traverse to node at pos - 1
  - Update link and delete node

Java Code:

```
public void deleteAtPosition(Node head, int pos) {
    if (head == null) return;
    if (pos == 0) {
        head = head.next;
        return;
    }
    Node current = head;
    for (int i = 0; current != null && i < pos - 1; i++) {
        current = current.next;
    }
    if (current == null || current.next == null) return;
    current.next = current.next.next;
}
```

3. Insertion in Doubly Linked List

Algorithm:

- Create a new node with data = x
- If pos == 0:
  - newNode.next = head
  - if head != null, head.prev = newNode
  - head = newNode
- Else:
  - Traverse to pos - 1
  - Insert node and adjust prev/next

Java Code:

```
class DNode {
    int data;
    DNode prev, next;
    DNode(int d) {
        data = d;
        prev = next = null;
    }
}

public void insertAtPosition(DNode head, int x, int pos) {
    DNode newNode = new DNode(x);
    if (pos == 0) {
        newNode.next = head;
        if (head != null) head.prev = newNode;
        head = newNode;
        return;
    }
    DNode current = head;
    for (int i = 0; current != null && i < pos - 1; i++) {
        current = current.next;
    }
    if (current == null) return;
    newNode.next = current.next;
    newNode.prev = current;
    if (current.next != null) current.next.prev = newNode;
    current.next = newNode;
}
```

4. Deletion in Doubly Linked List

Algorithm:

- If list is empty, return
- If pos == 0:
  - head = head.next
  - if head != null, head.prev = null
- Else:
  - Traverse to node at pos
  - Adjust prev and next
  - Delete node

Java Code:

```
public void deleteAtPosition(DNode head, int pos) {
    if (head == null) return;
    DNode current = head;
    if (pos == 0) {
        head = head.next;
        if (head != null) head.prev = null;
        return;
    }
    for (int i = 0; current != null && i < pos; i++) {
        current = current.next;
    }
    if (current == null) return;
    if (current.prev != null) current.prev.next = current.next;
    if (current.next != null) current.next.prev = current.prev;
}
```

5. Circular Linked List - Insertion and Deletion

Algorithm:

- In insertion:
  - If list is empty, new node points to itself
  - Else, traverse to last and adjust links
- In deletion:
  - If head is the only node, set head to null
  - Else, update last node's next and head

Java Code:

```
class CNode {
    int data;
    CNode next;
    CNode(int d) {
        data = d;
        next = null;
    }
}

public CNode insertEnd(CNode head, int x) {
    CNode newNode = new CNode(x);
    if (head == null) {
        newNode.next = newNode;
        return newNode;
    }
    CNode temp = head;
    while (temp.next != head) {
        temp = temp.next;
    }
    temp.next = newNode;
    newNode.next = head;
    return head;
}
```

```
public CNode deleteHead(CNode head) {
    if (head == null || head.next == head) return null;
    CNode last = head;
    while (last.next != head) {
        last = last.next;
    }
    last.next = head.next;
    head = head.next;
    return head;
}
```