EAST WEST UNIVERSITY

Assignment 1

Course:CSE 207

Sec:3

Submitted By:Nuran Farhana Prova

Student Id: 2023-1-60-075

Program Name: Object-Oriented Programming(Data Structure)

Date: 4/3/2024

Submitted to:

Dr. Anup Kumar Paul

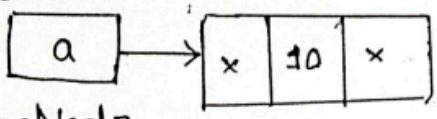
Associate professor,

Department of Computer Science and Engineering

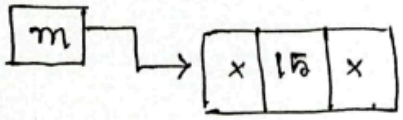
Insert At First:

Before Insert:

Start

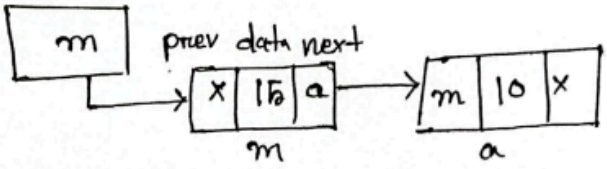


newNode



After Insert:

Start

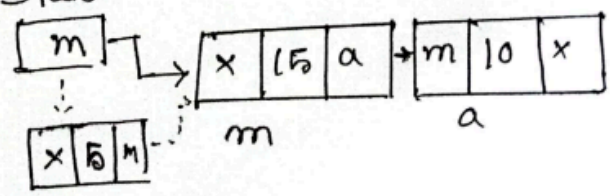


Algorithm:

```
newNode.next = Start;
Start.prev = newNode;
Start = newNode;
```

DeleteFirst:

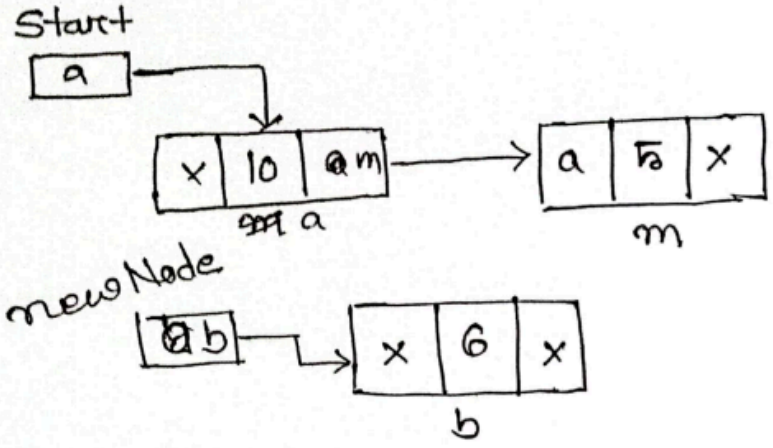
Start



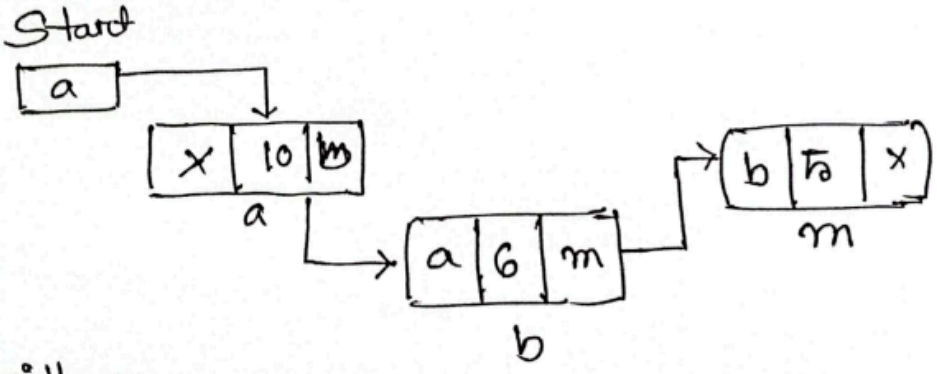
Algorithm:

```
Start = Start.next;
Start.prev = null;
```

Insert AT Middle:



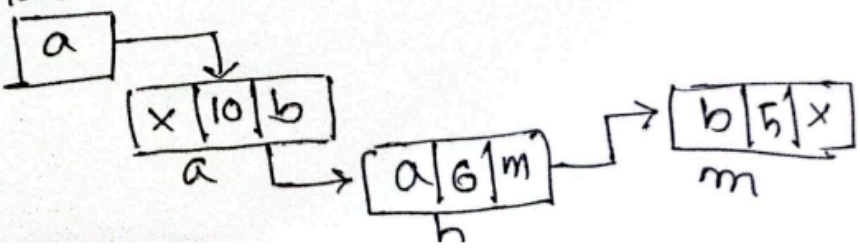
After Insert:



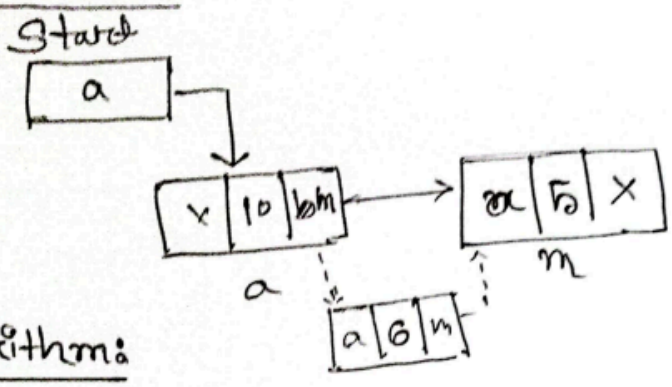
Algorithm:

```
newNode.prev = temp;
newNode.next = temp.next;
temp.next.prev = newNode;
temp.next = newNode;
```

DeleteMiddle:



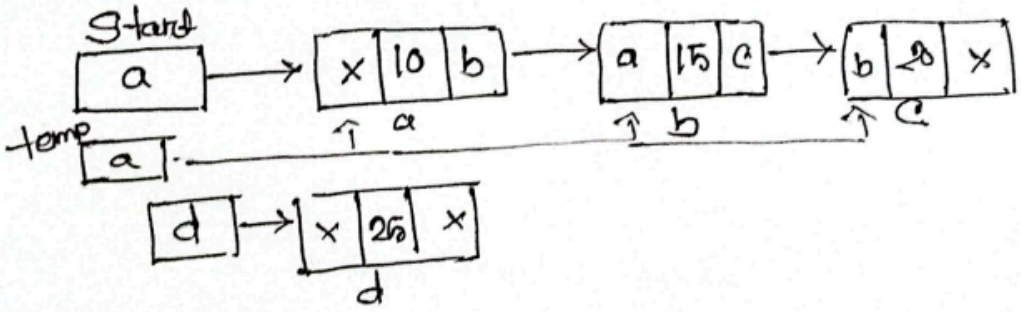
After Delete:



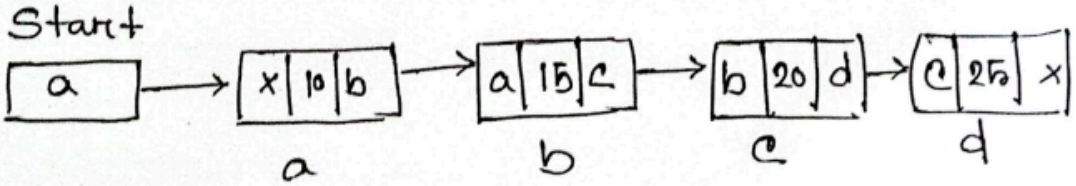
Algorithm:

```
temp.prev.next = temp.next;
temp.next.prev = temp.prev;
```

Insert At Last:



After Insert:



Algorithm:

```
temp.next = newNode;
newNode.prev = temp;
```

Delete Last:

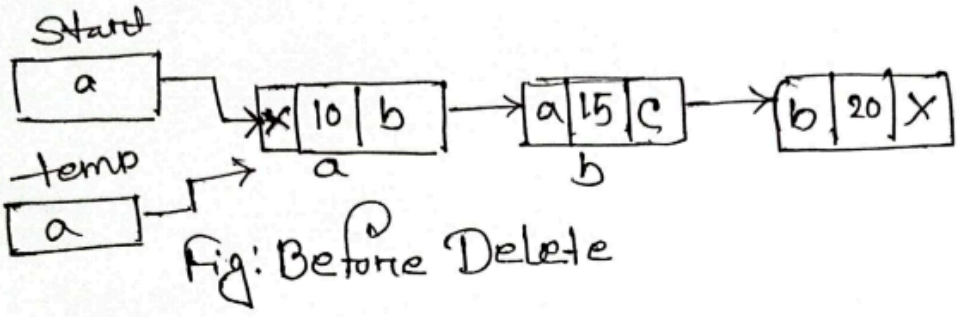


Fig: Before Delete

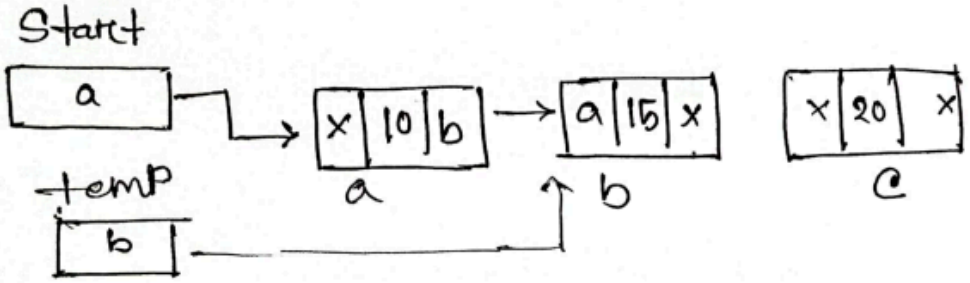


Fig: After Delete

Algorithm:

```
temp.prev.next = null;
```

Source Code:

```
package LinkedListOperation;
public class Node {
	int data;
	Node next;
	Node prev;
}
package LinkedListOperation;
import java.util.Scanner;
public class LinkedListOperations {
	Node start;
	public LinkedListOperations() {
		start = null;
	}
	 public Node getNode() {
		Node newNode = new Node();
		Scanner input = new Scanner(System.in);
		System.out.println("Enter the data: ");
		newNode.data = input.nextInt();
		newNode.next = null;
		newNode.prev = null;
		return newNode;
```

```
    }
 public void insertAtFirst() {
         Node newNode = getNode();
        if (start == null) {
           start = newNode;
        } else {
           newNode.next = start;
           start.prev = newNode;
           start = newNode;
        }
    }
 public void insertAtLast() {
        Node newNode = getNode();
if (start == null) {
   start = newNode;
} else {
   Node temp = start;
   while (temp.next != null) {
      temp = temp.next;
   }
   temp.next = newNode;
   newNode.prev = temp;
}
```

```
}
    public int nodeCounter() {
        Node temp = start;
        int count = 1;
        while(temp.next!=null) {
            temp = temp.next;
            count++;
        }
        return count;
    }
    public void insertAtMiddle() {
        Node newNode = getNode();
        Scanner input = new Scanner(System.in);
        System.out.print("\nEnter the position :");
        int position = input.nextInt();
        int totalNode = nodeCounter();
        if (position > 1 && position <= totalNode) {
            Node temp = start;
            int ctr = 1;
            while (ctr < position - 1) {
                temp = temp.next;
                ctr++;
            }
```

```
                newNode.prev = temp;
                newNode.next = temp.next;
                temp.next.prev = newNode;
                temp.next = newNode;
            } else {
                System.out.println("Invalid position");
            }
        }
    public void deleteFirst() {
            if (start == null) {
                System.out.println("Delete from start is not
possible");
            } else {
                start = start.next;
                start.prev = null;
            }
        }
    public void deleteLast() {
            if (start == null) {
        System.out.println("Delete from end is not
possible");
    } else {
        Node temp = start;
```

```
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.prev.next = null;
    }
}
    public void deleteMiddle() {

            Scanner input = new Scanner(System.in);
    System.out.print("\nEnter the delete position: ");
    int position = input.nextInt();
    int totalNode = nodeCounter();
    if (position > 1 && position < totalNode) {
        Node temp = start;
        int ctr = 1;
        while (ctr < position) {
            temp = temp.next;
            ctr++;
        }
        temp.prev.next = temp.next;
        temp.next.prev = temp.prev;
    } else {
        System.out.println("Invalid position");
```

```
    }
  }
      public void printList() {
            Node temp = start;
            while(temp!=null) {
                  if(temp.next!=null) {
                        System.out.print(temp.data+"-->");
                  }
                  else {
                        System.out.print(temp.data);
                  }
                  temp = temp.next;
            }
      }

}
```

Main Class:

```
package LinkedListOperation;
public class Main {
      public static void main(String[] args) {
            // TODO Auto-generated method stub
```

```
        LinkedListOperations list = new
LinkedListOperations();

        list.insertAtFirst();
        list.insertAtFirst();
        list.insertAtFirst();
        list.insertAtFirst();

        System.out.println("Before inserting at the
beginning");
        list.printList();
        list.insertAtLast();

        System.out.println("After inserting at the
beginning");
        list.printList();

        System.out.println("\nBefore inserting at the
middle");

        list.printList();

        list.insertAtMiddle();
```

```
        System.out.println("After inserting at the
middle");

        list.printList();

        System.out.println("\nBefore inserting at the
end");

        list.printList();

        list.insertAtLast();

        System.out.println("\nAfter inserting at the
end");

        list.printList();
        System.out.println("\nAfter delete first node");
        list.printList();

        System.out.println("\nBefore deleting at the
beginning");
```

```
            list.printList();
            list.deleteFirst();
            System.out.println("\nAfter deleting at the
beginning");
            list.printList();

            System.out.println("\nBefore deleting at the
middle");

            list.printList();
            list.deleteMiddle();
            System.out.println("After deleting at the
middle");
            list.printList();

            System.out.println("\nBefore deleting at the
end");

            list.printList();
            list.deleteLast();

            System.out.println("After deleting at the end");
```

```
            list.printList();
            System.out.println("\nFinally, traversal and
displaying the list:");
            list.printList();


        }
}
```

Output:

```
<terminated> Main (3) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (Mar 4, 2024, 1:24:06 AM – 1:25:07 AM) [pid: 15904]
10
Enter the data:
20
Enter the data:
30
Enter the data:
40
Before inserting at the beginning
40-->30-->20-->10Enter the data:
5
After inserting at the beginning
40-->30-->20-->10-->5
Before inserting at the middle
40-->30-->20-->10-->5Enter the data:
25

Enter the position :3
After inserting at the middle
40-->30-->25-->20-->10-->5
Before inserting at the end
40-->30-->25-->20-->10-->5Enter the data:
50

After inserting at the end
40-->30-->25-->20-->10-->5-->50
After delete first node
40-->30-->25-->20-->10-->5-->50
Before deleting at the beginning
40-->30-->25-->20-->10-->5-->50
After deleting at the beginning
30-->25-->20-->10-->5-->50
Before deleting at the middle
30-->25-->20-->10-->5-->50
Enter the delete position: 2
After deleting at the middle
30-->20-->10-->5-->50
Before deleting at the end
30-->20-->10-->5-->50After deleting at the end
30-->20-->10-->5
Finally, traversal and displaying the list:
30-->20-->10-->5
```