# Sukkur IBA University

## Data Structure Algorithm

**Name:** Tariq Mehmood

**CMS ID:** 023-23-017
**Semester:** 3rd

**Section:** E

**Submitted by:** Sir Riaz Hussain

**Lab No:** 03

**Q1:**

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

```
public class Task1{
    Node head;
    int size=0;
    void addFront(int data){
        size++;
        Node newNode=new Node(data);
        if(head==null){
            head=newNode;
            return;
        }
        newNode.next=head;
        head=newNode;
    }
    int getFrontItem(){
        if(head==null){
            System.out.println("Linked list is empty");

        }

        return head.data;
```

```
}
void removeFrontItem(){
        size--;
        if(head==null){
                System.out.println("Linked list is empty");
                return;
        }
        head=head.next;

}
void addToBack(int data){
        Node newNode=new Node(data);
        size++;
        Node temp=head;
        if(head==null){
                head=newNode;
                return;
        }
        else{
                while (temp.next!=null) {
```

```
                temp=temp.next;
            }
            temp.next=newNode;
        }
    }
    int getBackItem(){
        if(head==null){
            System.out.println("Linked list is empty");

        }
        Node temp=head;
        while (temp.next!=null) {
            temp=temp.next;
        }
        return temp.data;
    }
    void removeBackItem(){
        size--;
        Node temp=head;
        if(head==null){
            System.out.println("Linked list is empty");
            return;
        }
        while (temp.next.next!=null) {
            temp=temp.next;
        }
        temp.next=null;
    }
    boolean find(int data){

        Node temp=head;
        while (temp!=null) {
            if(temp.data==data){
                return true;
            }
            temp=temp.next;
        }
        return false;
    }
    void Remove(int key){
        size--;
        if(head==null){
            System.out.println("The Linked List is Empty ");
            return;
        }
        if(head.data==key){
            head=head.next;
            System.out.println(key+"  is removed from Linked 
List");
            return;
        }
        Node temp=head;
        Node prev=null;
        while (temp!=null) {
            if(temp.data==key){
                prev.next=temp.next;
                System.out.println(key+"  is removed from 
Linked List");
                return;
            }
```

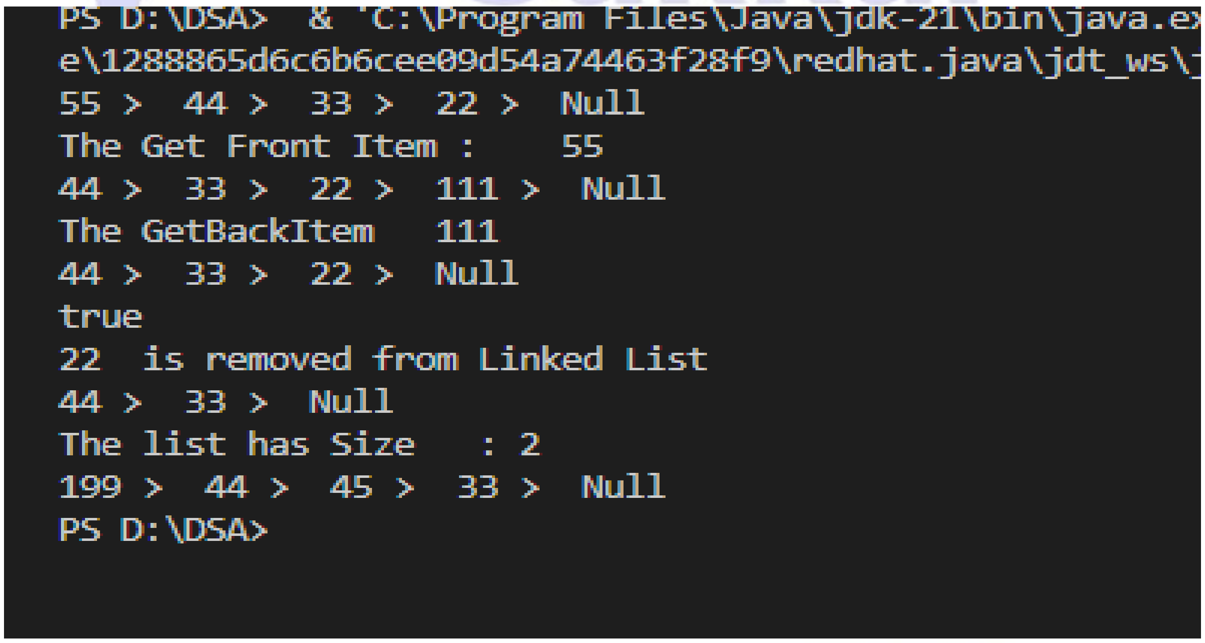
```
                prev=temp;
                temp=temp.next;
            }
            System.out.println(key+"  is not present in Linked
List");

        }
        void isListEmpty(){
            if(size==0){
                System.out.println("The List is empty");
            }
            else{
                System.out.println("The list has Size   : "+size);
            }
        }
        void addKeyBeforeNode(int key, int data){
            Node newNode=new Node(data);
            if(head==null){
                System.out.println("Key is not prsent, List is empty
Now add ");
                head=newNode;
                return;
          }
          if(head.data==key){
                newNode.next=head;
                head=newNode;
                return;
          }
          else {
          Node temp = head;
          while (temp.next != null) {
          if (temp.next.data == key) {
                newNode.next = temp.next;
                temp.next = newNode;
                return;
          }
          temp = temp.next;
          }
          System.out.println("Key is not present.");
          }
    }
    void addNodeAfterKey(int key, int data){
        Node newNode=new Node(data);
        if(head==null){
            System.out.println("Linked list is empty there is no
key here");
            head=newNode;
            return;
        }
        if(head.data==key){
            newNode.next=head.next;
            head.next=newNode;
            return;
        }

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


                temp.next.next=newNode;
```

```
                        return;
                    }


                    temp=temp.next;
                }
                System.out.println("Key " + key + " not found in the
list.");


            }
            void display(){
                Node temp=head;
                if(head==null){
                    System.out.println("Linked list is empty");


                }
                while (temp!=null) {
                    System.out.print(temp.data+" >  ");
                    temp=temp.next;
                }
                System.out.println("Null ");
```

```
	}
	public static void main(String[] args) {
		Task1 LinkedList=new Task1();
		LinkedList.addFront(22);
		LinkedList.addFront(33);
		LinkedList.addFront(44);
		LinkedList.addFront(55);
		LinkedList.display();
		LinkedList.addToBack(111);

		System.out.println("The Get Front Item : "+LinkedList.getFrontItem());
		LinkedList.removeFrontItem();
		LinkedList.display();
		System.out.println("The GetBackItem   "+ LinkedList.getBackItem());
		LinkedList.removeBackItem();
		LinkedList.display();
		System.out.println(LinkedList.find(33));
		LinkedList.Remove(22);
		LinkedList.display();
```

```
        LinkedList.isListEmpty();
        LinkedList.addKeyBeforeNode(44, 199);
        LinkedList.addNodeAfterKey(44, 45);
        LinkedList.display();



    }
}
```

**Ans:**



```
PS D:\DSA>  & 'C:\Program Files\Java\jdk-21\bin\java.ex
e\1288865d6c6b6cee09d54a74463f28f9\redhat.java\jdt_ws\
55 >  44 >  33 >  22 >  Null
The Get Front Item :    55
44 >  33 >  22 >  111 >  Null
The GetBackItem   111
44 >  33 >  22 >  Null
true
22  is removed from Linked List
44 >  33 >  Null
The list has Size   : 2
199 >  44 >  45 >  33 >  Null
PS D:\DSA>
```

**Q2:**

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

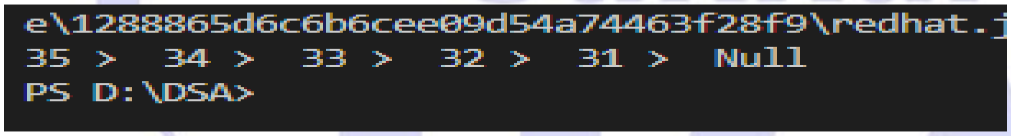
```
      Node next;

      Node (int data){

          this.data=data;

          this.next=null;

      }

}

/* here is Task 2 When we add in this Tail then  the effect only
/ AddtoBack() method not RemoveBack*/

public class Task2{

      Node head;

      Node tail;


      void addFront(int data){

            Node newNode=new Node(data);

            if(head==null){

                  head=newNode;

                  return;

            }

            newNode.next=head;

            head=newNode;
```

```
}
void addToBack(int data){
    Node newNode=new Node(data);
    if(head==null){
        head=newNode;
        tail=newNode;
        return;
    }
    else{
        tail.next=newNode;
        tail=newNode;
    }
}
void removeBackItem(){

    Node temp=head;
    if(head==null){
        System.out.println("Linked list is empty");
        return;
    }
```

```
        while (temp.next.next!=null) {
            temp=temp.next;
        }
        temp.next=null;
    }
    void display(){
        Node temp=head;
        if(head==null){
            System.out.println("Linked list is empty");

        }
        while (temp!=null) {
            System.out.print(temp.data+" >  ");
            temp=temp.next;
        }
        System.out.println("Null ");
    }
    public static void main(String[] args) {
      Task2 LL=new Task2();
      LL.addFront(31);
```

```
        LL.addFront(32);
        LL.addFront(33);
        LL.addFront(34);
        LL.addFront(35);

        LL.display();
    }
}
```

**Answer:**



```
e\1288865d6c6b6cee09d54a74463f28f9\redhat.j
35 >  34 >  33 >  32 >  31 >  Null
PS D:\DSA>
```

**Q3:**

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

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

```
}

public  class Task3 {
      Node head;
      int size=0;

      public int getSize() {
         return size;
      }
      public void insertAt(int index, int data) {
         if (index < 0 || index > size) {
            System.out.println("Invalid index.");
            return;
         }

         Node newNode = new Node(data);

         if (index == 0) {
            newNode.next = head;
            head = newNode;
```

```
        } else {
            Node current = head;
            for (int i = 0; i < index - 1; i++) {
                current = current.next;
            }
            newNode.next = current.next;
            current.next = newNode;
        }

        size++;
    }

    public int get(int index) {
        if (index < 0 || index >= size) {
            throw new IndexOutOfBoundsException("Index out of
bounds.");
        }

        Node current = head;
        for (int i = 0; i < index; i++) {
```

```
        current = current.next;
    }

    return current.data;
}

public void removeFrom(int index) {
    if (index < 0 || index >= size) {
        System.out.println("Invalid index.");
        return;
    }

    if (index == 0) {
        head = head.next;
    } else {
        Node current = head;
        for (int i = 0; i < index - 1; i++) {
            current = current.next;
        }
        current.next = current.next.next;
        }


        size--;
      }class Node {
    int data;
    Node next;

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

class Task4 {
    private Node head;

    Task4() {
        this.head = null;
    }
```

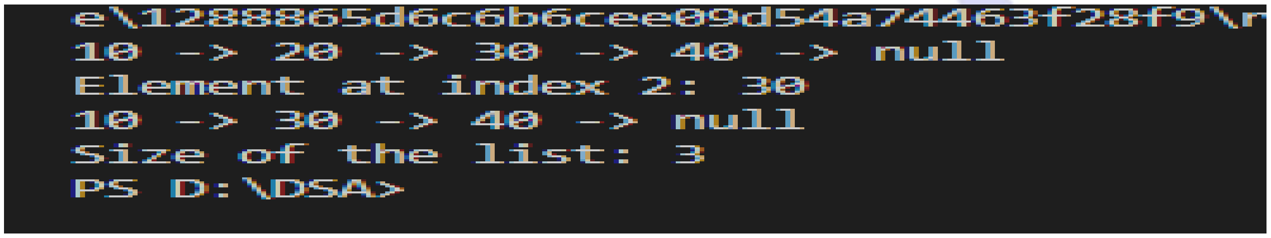
```
public void insertAtEnd(int data) {
    Node newNode = new Node(data);

    if (head == null) {
        head = newNode;
    } else {
        Node current = head;
        while (current.next != null) {
            current = current.next;
        }
        current.next = newNode;
    }
}

public void display() {
    if (head == null) {
        System.out.println("List is empty.");
        return;
    }
    Node current = head;
    while (current != null) {
        System.out.print(current.data + " -> ");
        current = current.next;
    }
    System.out.println("null");
}

public void reverse() {
    Node prev = null;
    Node current = head;
    Node next = null;

    while (current != null) {
        next = current.next;
        current.next = prev;
        prev = current;
        current = next;
    }
```

```
        head = prev;
    }

    public static void main(String[] args) {
        Task4 list = new Task4();

        list.insertAtEnd(10);
        list.insertAtEnd(20);
        list.insertAtEnd(30);
        list.insertAtEnd(40);

        System.out.println("Original List:");
        list.display();

        list.reverse();

        System.out.println("Reversed List:");
        list.display();
    }
}
```

```
public void display() {
    if (head == null) {
        System.out.println("List is empty.");
        return;
    }
    Node current = head;
    while (current != null) {
        System.out.print(current.data + " -> ");
        current = current.next;
    }
    System.out.println("null");
}

public static void main(String[] args) {
    Task3 list = new Task3();

    list.insertAt(0, 10);
    list.insertAt(1, 20);
    list.insertAt(2, 30);
```

```
        list.insertAt(3, 40);

        list.display();

        System.out.println("Element at index 2: " + list.get(2));

        list.removeFrom(1);
        list.display();

        System.out.println("Size of the list: " + list.getSize());
    }
  }
```

**Output:**



```
e\1288865d6c6b6cee09d54a74463f28f9\r
10 -> 20 -> 30 -> 40 -> null
Element at index 2: 30
10 -> 30 -> 40 -> null
Size of the list: 3
PS D:\DSA>
```

**Question 4;**

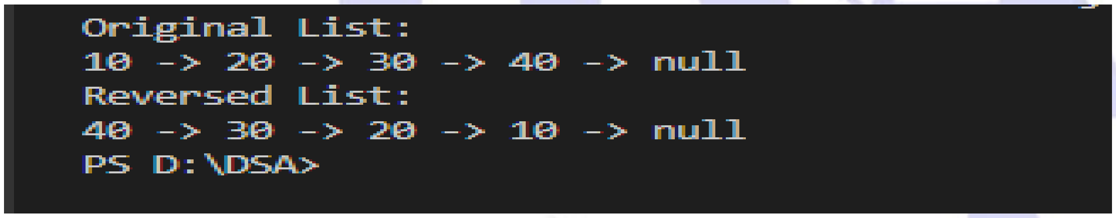
```
class Node {
   int data;
```

```
    Node next;

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

public class Task4 {
     Node head;

    public void insertAtEnd(int data) {
        Node newNode = new Node(data);

        if (head == null) {
            head = newNode;
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    public void display() {
        if (head == null) {
            System.out.println("List is empty.");
            return;
        }

        Node current = head;
        while (current != null) {
            System.out.print(current.data + " -> ");
            current = current.next;
        }
        System.out.println("null");
    }

    public void reverse() {
```

```
    Node prev = null;
    Node current = head;
    Node next = null;

    while (current != null) {
        next = current.next;
        current.next = prev;
        prev = current;
        current = next;
    }

    head = prev;
}

public static void main(String[] args) {
    Task4 list = new Task4();

    list.insertAtEnd(10);
    list.insertAtEnd(20);
    list.insertAtEnd(30);
```

```
        list.insertAtEnd(40);

        System.out.println("Original List:");
        list.display();

        list.reverse();

        System.out.println("Reversed List:");
        list.display();
    }
}
```

**Output**



```
Original List:
10 -> 20 -> 30 -> 40 -> null
Reversed List:
40 -> 30 -> 20 -> 10 -> null
PS D:\DSA>
```

**The End**