	codewitharrays.in freelance project available to buy contact on 8007592194	
SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

	freelance_Project available to buy contact on 8007592194	
21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+NextJs+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql

41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48	Train Ticket Booking Project	React+Springboot+MySql
49	Quizz Application Project	JSP+Springboot+MySql
50	Hotel Room Booking Project	React+Springboot+MySql

51	Online Crime Reporting Portal Project	React+Springboot+MySql
52	Online Child Adoption Portal Project	React+Springboot+MySql
53	online Pizza Delivery System Project	React+Springboot+MySql
54	Online Social Complaint Portal Project	React+Springboot+MySql
55	Electric Vehical management system Project	React+Springboot+MySql
56	Online mess / Tiffin management System Project	React+Springboot+MySql
57		React+Springboot+MySql
58		React+Springboot+MySql
59		React+Springboot+MySql
60		React+Springboot+MySql

Spring Boot + React JS + MySQL Project List

Sr.No	Project Name	YouTube Link
1	Online E-Learning Hub Platform Project	https://youtu.be/KMjyBaWmgzg?si=YckHuNzs7eC84-IW
2	PG Mate / Room sharing/Flat sharing	https://youtu.be/4P9cIHg3wvk?si=4uEsi0962CG6Xodp
3	Tour and Travel System Project Version 1.0	https://youtu.be/-UHOBywHaP8?si=KHHfE_A0uv725f12
4	Marriage Hall Booking	https://youtu.be/VXz0kZQi5to?si=llOS-QG3TpAFP5k7
5	Ecommerce Shopping project	https://youtu.be/vJ_C6LkhrZ0?si=YhcBylSErvdn7paq
6	Bike Rental System Project	https://youtu.be/FIzsAmIBCbk?si=7ujQTJqEgkQ8ju2H
7	Multi-Restaurant management system	https://youtu.be/pvV-pM2Jf3s?si=PgvnT-yFc8ktrDxB
8	Hospital management system Project	https://youtu.be/IynIouBZvY4?si=CXzQs3BsRkjKhZCw
9	Municipal Corporation system Project	https://youtu.be/cVMx9NVyI4I?si=qX0oQt-GT-LR_5jF
10	Tour and Travel System Project version 2.0	https://youtu.be/_4u0mB9mHXE?si=gDiAhKBowi2gNUKZ

Sr.No	Project Name	YouTube Link
11	Tour and Travel System Project version 3.0	https://youtu.be/Dm7nOdpasWg?si=P_Lh2gcOFhlyudug
12	Gym Management system Project	https://youtu.be/J8_7Zrkg7ag?si=LcxV51ynfUB7OptX
13	Online Driving License system Project	https://youtu.be/3yRzsMs8TLE?si=JRI_z4FDx4Gmt7fn
14	Online Flight Booking system Project	https://youtu.be/m755rOwdk8U?si=HURvAY2VnizIyJlh
15	Employee management system project	https://youtu.be/lD1iE3W_GRw?si=Y_jv1xV_BljhrD0H
16	Online student school or college portal	https://youtu.be/4A25aEKfei0?si=RoVgZtxMk9TPdQvD
17	Online movie booking system project	https://youtu.be/Lfjv_U74SC4?si=fiDvrhhrjb4KSlSm
18	Online Pizza Delivery system project	https://youtu.be/Tp3izreZ458?si=8eWAOzA8SVdNwlyM
19	Online Crime Reporting system Project	https://youtu.be/0UlzReSk9tQ?si=6vN0e70TVY1GOwPO
20	Online Children Adoption Project	https://youtu.be/3T5HC2HKyT4?si=bntP78niYH802I7N

Linked List-By Ashok Pate

22 April 2022 21:26

Program 1: Create Node and traverse that Linked list.

```
import java.util.Scanner;
//Create node and traversal of that node.
class LinkedList {

     static class Node{
        int data;
        Node next;
        public Node(int data){
            this.data=data;
            this.next=null;
        }
     }
    Node head=null;
    public  void createNode() {
        Scanner sc=new Scanner(System.in);
        int data,n;
        do {

            System.out.println("Enter data: ");
            data=sc.nextInt();
            Node new_node=new Node(data);
            if (head==null) {
                head=new_node;
            }
            else{
                new_node.next=head;
                head=new_node;
            }
            System.out.println("Do you want to add more data.if yes press 1");
            n=sc.nextInt();
        } while (n==1);
    }
    public void traversar() {
      Node temp=head;

        if(head==null){
          System.out.println("Linked list does not exist");
      }
      else{
          while(temp!=null){
            System.out.println(temp.data);
            temp=temp.next;
          }
      }
    }
    public static void main(String[] args) {
        LinkedList li=new LinkedList();
        li.createNode();
        li.traversar();

}
============================================================================
```

```
/*Singly Linked List complete code*/

import java.util.Iterator;
import java.util.Scanner;
//Create node and traversal of that node.
class Node {
    int data;
    Node next;
    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}
class LinkedList2 implements Iterable<Integer>, Iterator<Integer> {
    Node head;
```

```java
public boolean empty() {
    return this.head == null;
}
// Add node at first position logic 1
/*
 * public void addFirst(int data) {
 * Node newNode=new Node(data);
 * if(this.empty()){
 * this.head=newNode;
 * }
 * else{
 * newNode.next=this.head;
 * this.head=newNode;
 * }
 * }
 */
// Add node at first position logic 2
public void addFirst(int data) {
    Node newNode = new Node(data);
    if (!this.empty()) {
        newNode.next = this.head;
    }
    this.head = newNode;
}
// Add node at last Position
public void addLast(int data) {
    Node newNode = new Node(data);
    if (this.empty())
        this.head = newNode;
    else {
        Node trav = this.head;
        while (trav.next != null) {
            trav = trav.next;
        }
        trav.next = newNode;
    }
}
// find out node
public Node find(int data) {
    Node trav = this.head;
    while (trav != null) {
        if (trav.data == data)
            return trav;
        trav = trav.next;
    }
    return null;
}
// getcount of LinkedList
public int getcount() {
    int count = 0;
    Node trav = this.head;
    while (trav != null) {
        ++count;
        trav = trav.next;
    }
    return count;
}
// add node at perticular position
public void addAtPosition(int data, int position) {
    if (position <= 0 || position > (this.getcount() + 1)) {
        System.out.println("Invalid Position");
    }
    if (position == 1) {
        this.addFirst(data);
    } else if (position == this.getcount() + 1) {
        this.addLast(data);
    } else {
        Node trav = this.head;
        for (int count = 1; count < position - 1; count++) {
            trav = trav.next;
        }
```

```java
            Node newNode = new Node(data);
            newNode.next = trav.next;
            trav.next = newNode;
        }
    }
    // remove at first position
    public void removeFirst() {
        if (this.empty()) {
            System.out.println("LinkedList is Empty");
        }
        this.head = this.head.next;
    }
    // remove at last Postion
    public void removeLast() {
        if (this.empty()) {
            System.out.println("Linked list empty");
        }
        if (this.head.next == null) {
            this.head = null;
        } else {
            Node trav = this.head;
            while (trav.next.next != null) {
                trav = trav.next;
            }
            trav.next = null;
        }
    }
    // remove at perticular Position
    public void removeAtPosition(int position) {
        if (position <= 0 || position > this.getcount()) {
            System.out.println("Invalid Position");
        }
        if (position == 1) {
            this.removeFirst();
        } else if (position == this.getcount()) {
            this.removeLast();
        } else {
            Node trav = this.head;
            for (int count = 1; count < position - 1; count++) {
                trav = trav.next;
            }
            Node temp = trav.next;
            trav.next = temp.next;
            temp = null;
        }
    }
    // delete whole linked list
    public void clear() {
        while (!this.empty()) {
            this.removeFirst();
        }
    }
    // Display list
    public void display() {
        Node temp = this.head;
        if (this.empty()) {
            System.out.println("Linked list does not exist");
        } else {
            System.out.print("Head--> ");
            while (temp != null) {
                System.out.print(temp.data + "--> ");
                temp = temp.next;
            }
        }
        System.out.println("Null");
    }
    // reverse printing list
    public void reverse(Node head) {
        if (head == null)
            return;
        reverse(head.next);
```

```java
            System.out.print(head.data + "<--");
    }
    Node trav;
    @Override
    public Iterator<Integer> iterator() {
        this.trav = this.head;
        return this;
    }
    @Override
    public boolean hasNext() {
        return this.trav != null;
    }
    @Override
    public Integer next() {
        int data = this.trav.data;
        this.trav = this.trav.next;
        return null;
    }
}
public class LinkedList {
    private static Scanner sc = new Scanner(System.in);
    public static void acceptRecord(int[] data) {
        System.out.print("Enter data    :   ");
        data[0] = sc.nextInt();
    }
    public static void acceptPosition(int[] position) {
        System.out.print("Enter position    :   ");
        position[0] = sc.nextInt();
    }
    public static int menuList() {
        System.out.println("0.Exit");
        System.out.println("1.Add first.");
        System.out.println("2.Add last.");
        System.out.println("3.Add at position.");
        System.out.println("4.Remove first.");
        System.out.println("5.Remove last.");
        System.out.println("6.Remove from position.");
        System.out.println("7.Print List In Same Order");
        System.out.println("8.Print List In Reverse Order ");
        System.out.print("Enter choice  :   ");
        return sc.nextInt();
    }
    // public static void iterateLinkedList( LinkedList2 list ) {
    // for( int element : list )
    // System.out.print(element+" ");
    // System.out.println();
    // }
    public static void main(String[] args) {
        int choice;
        int[] data = new int[1];
        int[] position = new int[1];
        LinkedList2 list = new LinkedList2();
        while ((choice = LinkedList.menuList()) != 0) {
            switch (choice) {
                case 1:
                    LinkedList.acceptRecord(data);
                    list.addFirst(data[0]);
                    break;
                case 2:
                    LinkedList.acceptRecord(data);
                    list.addLast(data[0]);
                    break;
                case 3:
                    LinkedList.acceptRecord(data);
                    LinkedList.acceptPosition(position);
                    list.addAtPosition(data[0], position[0]);
                    break;
                case 4:
                    list.removeFirst();
                    break;
                case 5:
                    list.removeLast();
                    break;
                case 6:
                    LinkedList.acceptPosition(position);
                    list.removeAtPosition(position[0]);
                    break;
                case 7:
                    // Program.iterateLinkedList(list);
                    System.out.println("LL in same order: ");
                    list.display();
                    break;
                case 8:
                    System.out.println("Linked list in reverse order print: ");
                    list.reverse(list.head);
                    break;
                default:
                    System.out.println("Invalid choice");
                    break;
            }
        }
        list.clear();
    }
}
=====================================================================================================================

/*Singly Linked List using tail and Extra more advanced method for understanding */
class LL {
    private Node head;
    private Node tail;
    private int size;
    public LL() {
        this.size = 0;
    }
    private class Node {
        private int value;
        private Node next;
        public Node(int value) {
            this.value = value;
        }
        public Node(int value, Node next) {
            this.value = value;
            this.next = next;
        }
    }
    public void insertFirst(int val) {
        Node node = new Node(val);
        node.next = head;
        head = node;
        if (tail == null) {
            tail = head;
        }
        size += 1;
    }
    public void insertLast(int val) {
        if (tail == null) {
            insertFirst(val);
            return;
        }
        Node node = new Node(val);
        tail.next = node;
        tail = node;
        size++;
    }
    public void insert(int val, int index) {
        if (index == 0) {
            insertFirst(val);
            return;
        }
        if (index == size) {
            insertLast(val);
            return;
```

```java
        }
        Node temp = head;
        for (int i = 1; i < index; i++) {
            temp = temp.next;
        }
        Node node = new Node(val, temp.next);
        temp.next = node;
        size++;
    }
    // insert using recursion
    public void insertRec(int val, int index) {
        head = insertRec(val, index, head);
    }
    private Node insertRec(int val, int index, Node node) {
        if (index == 0) {
            Node temp = new Node(val, node);
            size++;
            return temp;
        }
        node.next = insertRec(val, index-1, node.next);
        return node;
    }
    public int deleteFirst() {
        int val = head.value;
        head = head.next;
        if (head == null) {
            tail = null;
        }
        size--;
        return val;
    }
    public int deleteLast() {
        if (size <= 1) {
            return deleteFirst();
        }
        Node secondLast = get(size - 2);
        int val = tail.value;
        tail = secondLast;
        tail.next = null;
        size--;
        return val;
    }
    public int delete(int index) {
        if (index == 0) {
            return deleteFirst();
        }
        if (index == size - 1) {
            return deleteLast();
        }
        Node prev = get(index - 1);
        int val = prev.next.value;
        prev.next = prev.next.next;
        size--;
        return val;
    }
    public Node find(int value) {
        Node node = head;
        while (node != null) {
            if (node.value == value) {
                return node;
            }
            node = node.next;
        }
        return null;
    }
    public Node get(int index) {
        Node node = head;
        for (int i = 0; i < index; i++) {
            node = node.next;
        }
        return node;
    }
    public void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.value + " -> ");
            temp = temp.next;
        }
        System.out.println("END");
    }
    // https://leetcode.com/problems/remove-duplicates-from-sorted-list
    public void duplicates() {
        Node node = head;
        while (node.next != null) {
            if (node.value == node.next.value) {
                node.next = node.next.next;
                size--;
            } else {
                node = node.next;
            }
        }
        tail = node;
        tail.next = null;
    }
    // https://leetcode.com/problems/merge-two-sorted-lists/submissions/
    public static LL merge(LL first, LL second) {
        Node f = first.head;
        Node s = second.head;
        LL ans = new LL();
        while (f != null && s != null) {
            if (f.value < s.value) {
                ans.insertLast(f.value);
                f = f.next;
            } else {
                ans.insertLast(s.value);
                s = s.next;
            }
        }
        while (f != null) {
            ans.insertLast(f.value);
            f = f.next;
        }
        while (s != null) {
            ans.insertLast(s.value);
            s = s.next;
        }
        return ans;
    }
    // recursion reverse
    private void reverse(Node node) {
        if (node == tail) {
            head = tail;
            return;
        }
        reverse(node.next);
        tail.next = node;
        tail = node;
        tail.next = null;
    }
//https://leetcode.com/problems/reverse-linked-list/
    public void reverse() {
        if (size < 2) {
            return;
        }
        Node prev = null;
        Node present = head;
        Node next = present.next;
        while (present != null) {
            present.next = prev;
            prev = present;
            present = next;
            if (next != null) {
                next = next.next;
            }
        }
        head = prev;
    }
    public static void main(String[] args) {
        LL list = new LL();
        System.out.println("ADD first zale 3 ,2,8,17");
        list.insertFirst(3);
        list.insertFirst(2);
        list.insertFirst(8);
        list.insertFirst(17);
        list.display();
        System.out.println("Add Last 99 zala");
        list.insertLast(99);
        list.display();
        System.out.println("Insert at perticular 100 zala");
        list.insert(100, 3);
        list.display();
        System.out.println("delete first zala");
        System.out.println(list.deleteFirst());
        list.display();
        System.out.println("delete last zala");
        System.out.println(list.deleteLast());
        list.display();
        System.out.println("delete at perticular zala");
        System.out.println(list.delete(2));
        list.display();
        System.out.println("insert using recursion 88 kel");
        list.insertRec(88, 2);
        list.display();
        System.out.println("Parat first la add kele 3,3,5,6,7,7");
        list.insertFirst(3);
        list.insertFirst(3);
        list.insertFirst(5);
        list.insertFirst(5);
        list.insertLast(6);
        list.insertLast(7);
        list.insertLast(7);
        list.display();
        System.out.println("duplicates la called kela");
        list.duplicates();
        list.display();
        LL first = new LL();
        LL second = new LL();
        System.out.println("ek New LL banvli 1,3,5");
        first.insertLast(1);
        first.insertLast(3);
        first.insertLast(5);
        list.display();
        System.out.println("dusri LL banvali 1,2,9,14");
        second.insertLast(1);
        second.insertLast(2);
        second.insertLast(9);
        second.insertLast(14);
        list.display();
        System.out.println("Doghana merge krun takl");
        LL ans = LL.merge(first, second);
        ans.display();
    }
}
/*Output Of the above program for observation:-
ADD first zale 3 ,2,8,17
17 -> 8 -> 2 -> 3 -> END
Add Last 99 zala
17 -> 8 -> 2 -> 3 -> 99 -> END
Insert at perticular 100 zala
17 -> 8 -> 2 -> 100 -> 3 -> 99 -> END
delete first zala
17
8 -> 2 -> 100 -> 3 -> 99 -> END
```

```
delete last zala
99
8 -> 2 -> 100 -> 3 -> END
delete at perticular zala
100
8 -> 2 -> 3 -> END
insert using recursion 88 kel
8 -> 2 -> 88 -> 3 -> END
Parat first la add kele 3,3,5,6,7,7
5 -> 5 -> 3 -> 3 -> 8 -> 2 -> 88 -> 3 -> 6 -> 7 -> 7 -> END
duplicates la called kela
5 -> 3 -> 8 -> 2 -> 88 -> 3 -> 6 -> 7 -> END
ek New LL banvli 1,3,5
5 -> 3 -> 8 -> 2 -> 88 -> 3 -> 6 -> 7 -> END
dusri LL banvali 1,2,9,14
5 -> 3 -> 8 -> 2 -> 88 -> 3 -> 6 -> 7 -> END
Doghana merge krun takl
1 -> 1 -> 2 -> 3 -> 5 -> 9 -> 14 -> END
*/
=============================================================================================


/*Singly Linked List Add value after Data & Add value Before Data Method
                    Delete value after Data & Delete value Before Data Method
                    and find index of particular data element */
class Node {
    int data;
    Node next;
    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}
class InnerTest1 {
    Node head;
    public boolean empty() {
        return this.head == null;
    }
    public void addFirst(int data) {
        Node newNode = new Node(data);
        if (!this.empty()) {
            newNode.next = this.head;
        }
        this.head = newNode;
    }
    public void display() {
        Node temp = this.head;
        if (this.empty()) {
            System.out.println("Linked list does not exist");
        } else {
            System.out.print("Head--> ");
            while (temp != null) {
                System.out.print(temp.data + "--> ");
                temp = temp.next;
            }
        }
        System.out.println("Null");
    }
    public void addAfterValue(int afterdata, int data) {
        Node temp = head;
        Node newNode = new Node(data);
        while (temp != null) {
            if (temp.data == afterdata) {
                newNode.next = temp.next;
                temp.next = newNode;
                break;
            } else {
                temp = temp.next;
            }
        }
    }
```

```java
    public void addBeforeValue(int data, int beforedata) {
        Node newNode = new Node(data);
        Node temp = head;
        while (temp != null) {
            if (temp.data == beforedata) {
                Node pre = getPrevious(temp);
                newNode.next = temp;
                pre.next = newNode;
                break;
            }
            temp = temp.next;
        }
    }
    public Node getPrevious(Node newNode) {
        Node temp = head;
        while (temp != null) {
            if (temp.next == newNode) {
                return temp;
            }
            temp = temp.next;
        }
        return null;
    }
    public void removeAfterValue(int data){
        var current = head;
        while (current != null){
            if (current.data == data){
                current.next = current.next.next;
            }
            current = current.next;
        }

    }
    public void removeBeforeValue(int data){
        var current = head;
        while (current != null){
            if (current.data == data){
                var previous = getPrevious1(current);
                var previousPre = getPrevious(previous);
                previousPre.next = current;
            }
            current = current.next;
        }
    }
    private Node getPrevious1(Node node){
        var current = head;
        while (current != null){
            if (current.next == node){
                return current;
            }
            current = current.next;
        }
        return null;
    }
    public int getIndexOf(int data){
        if (head == null){
            return -1;
        }
        var current = head;
        int index = 0;
        while (current != null){
            if (current.data == data){
                return index;
            }
            current = current.next;
            index++;
        }
        return -1;
    }
}
public class Test1 {
    public static void main(String[] args) {
        InnerTest1 t1 = new InnerTest1();
        System.out.println("First add kele 10,20,30,40");
        t1.addFirst(10);
        t1.addFirst(20);
        t1.addFirst(30);
        t1.addFirst(40);
        t1.display();
        System.out.println("30 ntr 50 add kela");
        t1.addAfterValue(30, 50);
        t1.display();
        System.out.println("50 chya adhi 60 add kela");
        t1.addBeforeValue(60, 50);
        t1.display();
        System.out.println("20 ntr delete kela");
        t1.removeAfterValue(20);
        t1.display();
        System.out.println("50 chya adhi delete kela");
        t1.removeBeforeValue(50);
        t1.display();
        System.out.println("50 cha index find kela");
        int result=t1.getIndexOf(50);
        System.out.println("Index: "+result);

    }
}
/** Output for oservation:-
 First add kele 10,20,30,40
Head--> 40--> 30--> 20--> 10--> Null
30 ntr 50 add kela
Head--> 40--> 30--> 50--> 20--> 10--> Null
50 chya adhi 60 add kela
Head--> 40--> 30--> 60--> 50--> 20--> 10--> Null
20 ntr delete kela
Head--> 40--> 30--> 60--> 50--> 20--> Null
50 chya adhi delete kela
Head--> 40--> 30--> 50--> 20--> Null
50 cha index find kela
Index: 2
 */
=============================================================================
```

https://www.youtube.com/@codewitharrays

https://www.instagram.com/codewitharrays/

https://t.me/codewitharrays Group Link: https://t.me/cceesept2023

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project