```java
import java.util.Scanner;

class LinkListDS
{
    class Node {
        int data;
        Node next;

        Node(int d) {
            data = d;
            next = null;
        }
    }
        Node root;

        LinkListDS()
        {
            root = null;
        }
        public void insertNode(int e) {
            Node n = new Node(e);
            if (root == null) {
                root = n;
            } else {
                n.next = root;
                root = n;
            }
        }


        public void deleteNode() {
                if (root == null) {
                    System.out.println("Link List Empty");
                } else {
                    Node t = root;
                    root = root.next;
                    System.out.println(t.data + "Remove");
                }
            }

        public void printLink() {
            if (root == null) {
                System.out.println("Empty List");
            } else {
                Node t = root;
                while (t != null) {

                    System.out.println(t.data);
                    t = t.next;
                }
            }

        }
        public void sortList(Stack a) {
            if (root==null) {
                System.out.println("List is Empty");
            } else {
                Node t = root;
                while (t != null) {
                    a.push(t.data);
                    t = t.next;
                }
```

```
                Node s = root;
                while (s != null) {
                    s.data = a.pop();
                    s = s.next;
                }
            }
            System.out.println();
        }
}

class Stack {
     int s[];
     int top;
     int maxsize;

     Stack() {

    }

    public Stack(int size) {
        maxsize = size;
        s = new int[maxsize];
        top = -1;
    }


    public void push(int data) {
        top++;
        s[top] = data;
    }

    public int pop() {
        int poped = s[top];
        top--;
        return poped;
    }

    public int peek() {
        return s[top];
    }

    public void print() {
        System.out.print("Stack :");
        for (int i = top; i > -1; i--) {
            System.out.print(s[i] + " ");
        }
        System.out.println();
    }
}
public class Reverce {
    public static void main(String args[]) {
        int val, ch;
        Scanner s = new Scanner(System.in);
        LinkListDS q = new LinkListDS();
        Stack t=new Stack();
    do {
            System.out.println(
                    "\n1.Insert Node  \n2.Delete Node \n3.Pirnt List \nEnter choice :");
            ch = s.nextInt();

            switch (ch) {
            case 1:
```

```
125                 System.out.println("Enter Left Node");
126                 val = s.nextInt();
127                 q.insertNode(val);
128                 break;
129             case 2:
130                 System.out.println("Enter Right Node");
131                 q.deleteNode();
132                 break;
133             case 3:
134                 System.out.println("LinkList");
135                 q.printLink();
136                 System.out.println("Rev");
137                 q.sortList(t);
138                 break;
139             }
140 
141     }while(ch!=0);
142     }
143 }
144 
```

```java
import java.util.Scanner;
class LinkList
{
    class Node {
        int data;
        Node next;

        Node(int d) {
            data = d;
            next = null;
        }
    }
        Node root;

        LinkList()
        {
            root = null;
        }
        public void insertNode(int e) {
            Node n = new Node(e);
            if (root == null) {
                root = n;
            } else {
                n.next = root;
                root = n;
            }
        }


        public void deleteNode() {
                if (root == null) {
                    System.out.println("Link List Empty");
                } else {
                    Node t = root;
                    root = root.next;
                    System.out.println(t.data + "Remove");
                }
            }

        public void sortLink()
        {
            if (root == null) {
                System.out.println("Empty List");
            }
            else {
                int temp=-1;
             for(Node p=root;p!=null;p=p.next)
             {
                 for(Node t=root ,t2=root.next;t!=null && t2!=null;t=t.next,t2=t2.next)
                 {
                    if(t.data>t2.data)
                    {
                         temp=t.data;
                        t.data=t2.data;
                        t2.data=temp;

                    }


                 }
             }
             System.out.println();
```

```
63              }
64
65          }
66
67          public void printLink() {
68              if (root == null) {
69                  System.out.println("Empty List");
70              } else {
71                  Node t = root;
72                  while (t != null) {
73
74                      System.out.println(t.data);
75                      t = t.next;
76                  }
77              }
78
79          }
80
81      public Node Start() {
82              return (root == null) ? null : root;
83          }
84  }
85
86
87  public class MergeList
88  {
89
90      public static void main(String args[]) {
91          int val, ch;
92          Scanner s = new Scanner(System.in);
93          LinkList first = new LinkList();
94          LinkList second = new LinkList();
95          LinkList third = new LinkList();
96
97          do {
98              System.out.println("\n1.First List Node \n2.Second List Node \n3.Delete First
    Node \n4.Delete Second Node \n5.Print Merged LinkList");
99              ch = s.nextInt();
100             switch (ch) {
101             case 1:
102                 System.out.println("Enter First List Node");
103                 val = s.nextInt();
104                 first.insertNode(val);
105                 break;
106
107             case 2:
108                 System.out.println("Enter Second List Node");
109                 val = s.nextInt();
110                 second.insertNode(val);
111                 break;
112
113             case 3:
114                 System.out.println("Delete First Node");
115                 first.deleteNode();
116                 break;
117             case 4:
118                 System.out.println("Delete Second Node");
119                 second.deleteNode();
120                 break;
121             case 5:
122                 System.out.println("Print Merge LinkList-->");
123
```

```
124                 System.out.println("First LinkList\n");
125                 first.printLink();
126                 System.out.println("Second LinkList\n");
127                 second.printLink();
128 
129                 LinkList.Node a, b;
130 
131                 a = first.Start();
132                 b = second.Start();
133 
134                 while (a != null && b != null) {
135                     if (a.data < b.data) {
136                         third.insertNode(a.data);
137                         a = a.next;
138 
139                     } else {
140                         third.insertNode(b.data);
141                         b = b.next;
142                     }
143                 }
144                 if (a != null) {
145                     while (a != null) {
146                         third.insertNode(a.data);
147                         a = a.next;
148                     }
149                 }
150                 if (b != null) {
151                     while (b != null) {
152                         third.insertNode(b.data);
153                         b = b.next;
154                     }
155                 }
156 
157                 System.out.println("Print Merged LinkList\n");
158                 third.printLink();
159                 break;
160             }
161 
162         } while (ch != 0);
163 }
164     }
165 
166 
167 
```

```java
import java.util.Scanner;
class List
{
    class Node {
        int data;
        Node next;

        Node(int d) {
            data = d;
            next = null;
        }
    }
        Node root;

        List()
        {
            root = null;
        }
        public void insertNode(int e) {
            Node n = new Node(e);
            if (root == null) {
                root = n;
            } else {
                n.next = root;
                root = n;
            }
        }


        public void deleteNode() {
                if (root == null) {
                    System.out.println("Link List Empty");
                } else {
                    Node t = root;
                    root = root.next;
                    System.out.println(t.data + "Remove");
                }
            }

        public void sortLink()
        {
            if (root == null) {
                System.out.println("Empty List");
            }
            else {
                int temp=-1;
             for(Node p=root;p!=null;p=p.next)
             {
                 for(Node t=root ,t2=root.next;t!=null && t2!=null;t=t.next,t2=t2.next)
                 {
                    if(t.data>t2.data)
                    {
                         temp=t.data;
                        t.data=t2.data;
                        t2.data=temp;

                    }


                 }
             }
             System.out.println();
```

```
63              }
64
65          }
66
67          public void printLink() {
68              if (root == null) {
69                  System.out.println("Empty List");
70              } else {
71                  Node t = root;
72                  while (t != null) {
73
74                      System.out.println(t.data);
75                      t = t.next;
76                  }
77              }
78
79          }
80
81  }
82
83
84  public class SortList
85  {
86
87      public static void main(String args[]) {
88          int val, ch;
89          Scanner s = new Scanner(System.in);
90          List q = new List();
91
92          do {
93              System.out.println("\n1.Insert Node \n2.Delete Node \n3.Print Link List \n
    Enter choice :");
94              ch = s.nextInt();
95              switch (ch) {
96              case 1:
97                  System.out.println("Enter Left Node");
98                  val = s.nextInt();
99                  q.insertNode(val);
100                 break;
101
102             case 2:
103                 System.out.println("Enter Delete Left Node");
104                 q.deleteNode();
105                 break;
106
107             case 3:
108                 System.out.println("Befor Sorting");
109                 q.printLink();
110                 q.sortLink();
111                 System.out.println("After Sorting");
112                 q.printLink();
113                 break;
114             }
115         } while (ch != 0);
116     }
117
118 }
119
```