Assignment 4

Linked List Questions

Q.1 : Find the middle element in Linked List

if the given linked list is 1->2->3->4->5->6 then the output should be 4.Sol :

Code - >

```
import java.util.Scanner;


public class Assignment {

        public static class Node {

            Node next;
            int data;

            public Node(int data) {
                this.data = data;

            }
        }

                public static void printMiddle(Node head) {
                Node fast = head;
                Node slow = head;
                while (fast != null && fast.next != null) {
                    slow = slow.next;
                    fast = fast.next.next;
                }
                System.out.println(slow.data);
            }



        // Driver code
        public static void main(String[] args)
        {

            Scanner sc = new Scanner(System.in);
            Node head = new  Node (1);
            Node node1 = new  Node (2);
```

```
            Node node2 = new  Node (3);
            Node node3 = new  Node (4);
            Node node4 = new  Node (5);
            Node node5 = new  Node (6);
            head.next=node1;
            node1.next=node2;
            node2.next=node3;
            node3.next=node4;
            node4.next=node5;

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

            }
            System.out.println();
            System.out.println("Middle of Linked list is");
             printMiddle(head);

        }
    }
```

Output ->

1->2->3->4->5->6

Middle of Linked list is

4

Process finished with exit code 0

Q.2 : Sort the LinkedList.

if the given linked list is 1->4->2->5->3->6 then the output should be 1->2->3->4->5->6

Sol :

Code - >

```
import java.util.ArrayList;
```

```
public class CopyArrayList {
   public static void main(String[] args) {
import java.util.Scanner;


public class Assignment {

        public static class Node {

            Node next;
            int data;

            public Node(int data) {
                this.data = data;

            }
        }


    public static int lengthLL(Node head){
        int count = 1;
        while(head.next != null){
            head = head.next;
            count++;
        }
        return count;
    }

    public static Node sortList(Node head )
    {     if(head==null || head.next==null)
        return head;
        for(int i=0;i<lengthLL(head)-1;i++){
            Node prev = null;
            Node curr = head;
            Node next = curr.next;

            while(curr.next != null){
                if(curr.data > curr.next.data){
                    if(prev == null){
                        curr.next = next.next;
                        next.next = curr;
                        prev = next;
                        head = prev;
                    }else{
                        next = curr.next;
                        curr.next = next.next;
                        prev.next = next;
                        next.next = curr;
                        prev = next;
                    }
                }else{
```

```
                    prev = curr;
                    curr = curr.next;
                }
            }
        }
        return head;

    }

        // Driver code
        public static void main(String[] args)
        {

            Scanner sc = new Scanner(System.in);
            Node head = new  Node (1);
            Node node1 = new  Node (4);
            Node node2 = new  Node (2);
            Node node3 = new  Node (5);
            Node node4 = new  Node (3);
            Node node5 = new  Node (6);
            head.next=node1;
            node1.next=node2;
            node2.next=node3;
            node3.next=node4;
            node4.next=node5;

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

            }
            System.out.println();
            System.out.println("Sorted Linked list is");
            Node newHead = sortList(head);


            Node currNew = newHead;
            while (currNew!= null){
                if (currNew.next == null){
                    System.out.print(currNew.data);
                } else {
                    System.out.print(currNew.data + "->");
                }
                currNew=currNew.next;

            }
```

```
        }
    }
```

Output ->

1->4->2->5->3->6

Sorted Linked list is

1->2->3->4->5->6

Process finished with exit code 0

Q.3 : Reverse the LinkedList.

if the given linked list is 1->2->3->4->5->6 then the output should be 6->5->4->3->2->1

Sol :

Code - >

```
import java.util.Scanner;


public class Assignment {

        public static class Node {

            Node next;
            int data;

            public Node(int data) {
                this.data = data;

            }
        }
        public static void printReverse(Node root) {

        if(root == null){
            return;
        }
        printReverse(root.next);
        System.out.print(root.data+" ");


    }
```

```
        // Driver code
        public static void main(String[] args)
        {

            Scanner sc = new Scanner(System.in);
            Node head = new  Node (1);
            Node node1 = new  Node (2);
            Node node2 = new  Node (3);
            Node node3 = new  Node (4);
            Node node4 = new  Node (5);
            Node node5 = new  Node (6);
            head.next=node1;
            node1.next=node2;
            node2.next=node3;
            node3.next=node4;
            node4.next=node5;
           // node5.next=node2;

            Node curr = head;
            while (curr!= null){
                if (curr.next == null){
                    System.out.print(curr.data);
                } else {
                    System.out.print(curr.data + " ");
                }
                curr=curr.next;

            }
            System.out.println();
            System.out.println("Reversed Linked list is");
            printReverse(head);

        }
    }
```

Output ->

1 2 3 4 5 6

Reversed Linked list is

6 5 4 3 2 1

Process finished with exit code 0

Q.4 : Detect that linked list has a loop or not.

if loop is there return or print true

if not return or print false.

Sol :

Code - >

```
import java.util.Scanner;


public class Assignment {

        public static class Node {

            Node next;
            int data;

            public Node(int data) {
                this.data = data;

            }
        }
        public static boolean detectLoop(Node head) {
        Node fast = head;
        Node slow = head;
        while (fast != null &&  fast.next != null) {

            slow = slow.next;
            fast = fast.next.next;
            if (fast == slow){
                return true;
            }
        }

       return false;
    }



        // Driver code
        public static void main(String[] args)
        {

            Scanner sc = new Scanner(System.in);
            Node head = new  Node (1);
            Node node1 = new  Node (2);
            Node node2 = new  Node (3);
            Node node3 = new  Node (4);
            Node node4 = new  Node (5);
            Node node5 = new  Node (6);
            head.next=node1;
            node1.next=node2;
```

```
            node2.next=node3;
            node3.next=node4;
            node4.next=node5;
            node5.next=node2;

            boolean isLoop = detectLoop(head);

            System.out.println(isLoop);


            Node head2 = new  Node (10);
            Node node1a = new  Node (20);
            Node node2b = new  Node (30);
            Node node3c = new  Node (40);

            head2.next=node1a;
            node1a.next=node2b;
            node2b.next=node3c;


            boolean isLoopNew = detectLoop(head2);

            System.out.println(isLoopNew);


        }
    }
```

Output ->

true

false

Process finished with exit code 0