Linked list Assignment

Answer to question no.1

Java code

```
class LinkedList {
 Node head;



 static class Node {
     int data;
     Node next;

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



 public boolean search(int X) {
     Node current = head;
     while (current != null) {
         if (current.data == X)
             return true;
         current = current.next;
     }
     return false;
 }

 public static void main(String[] args)
 {
     LinkedList list1 = new LinkedList();
     LinkedList list2 = new LinkedList();



     list1.head = new Node(14);
     list1.head.next = new Node(21);
     list1.head.next.next = new Node(11);
     list1.head.next.next.next = new Node(30);
     list1.head.next.next.next.next = new Node(10);
```

```
        list2.head = new Node(6);
        list2.head.next = new Node(21);
        list2.head.next.next = new Node(17);
        list2.head.next.next.next = new Node(30);
        list2.head.next.next.next.next = new Node(10);
        list2.head.next.next.next.next.next = new Node(8);




        int X = 14,Y = 13;

        if (list1.search(X))
            System.out.println("Yes");
        else
            System.out.println("No");

        if (list2.search(Y))
            System.out.println("Yes");
        else
            System.out.println("No");




    }
}
```

Output :

```java
class LinkedList {
    public static void main(String[] args)
        list1.head.next.next.next = new Node(data:30);
        list1.head.next.next.next.next = new Node(data:10);

        list2.head = new Node(data:6);
        list2.head.next = new Node(data:21);
        list2.head.next.next = new Node(data:17);
        list2.head.next.next.next = new Node(data:30);
        list2.head.next.next.next.next = new Node(data:10);
        list2.head.next.next.next.next.next = new Node(data:8);



        int X = 14,Y = 13;

        if (list1.search(X))
            System.out.println(x:"Yes");
        else
            System.out.println(x:"No");

        if (list2.search(Y))
            System.out.println(x:"Yes");
        else
            System.out.println(x:"No");



    }
}
```

```
PS C:\Users\HP>  & 'C:\Program Files\Java\jdk-22\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\HP\AppData\Local
jdt.ls-java-project\bin' 'LinkedList'
Yes
No
PS C:\Users\HP>
```

Answer to question no: 2

```java
class LinkedList {
    Node head;



    static class Node {
        int data;
        Node next;

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



    public void insertAfter(Node prev_node, int new_data) {
        if (prev_node == null) {
```

```
            System.out.println("Previous node cannot be null");
            return;
        }

        Node new_node = new Node(new_data);
        new_node.next = prev_node.next;
        prev_node.next = new_node;
    }



    public void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " -> ");
            temp = temp.next;
        }
        System.out.println("NULL");
    }

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

        list.head = new Node(1);
        list.head.next = new Node(2);
        list.head.next.next = new Node(4);
        list.head.next.next.next = new Node(5);
        list.head.next.next.next.next = new Node(6);

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



        list.insertAfter(list.head.next, 3);

        System.out.println("Linked List after insertion:");
        list.printList();
    }
}
```

Output:

```
File  Edit  Selection  View  Go  Run  Terminal  Help
libmwlmgrimpl.dll | insert.java 1 | class LinkedList { Untitled-1 | Settings
C: > Users > HP > insert.java > LinkedList
class LinkedList {
    public void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " -> ");
            temp = temp.next;
        }
        System.out.println(x:"NULL");
    }

    Run | Debug
    public static void main(String[] args) {
        LinkedList list = new LinkedList();

        list.head = new Node(data:1);
        list.head.next = new Node(data:2);
        list.head.next.next = new Node(data:4);
        list.head.next.next.next = new Node(data:5);
        list.head.next.next.next.next = new Node(data:6);

        System.out.println(x:"Original Linked List:");
        list.printList();


        list.insertAfter(list.head.next, new_data:3);

        System.out.println(x:"Linked List after insertion:");
        list.printList();
    }
}

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

7320\jdt_ws\jdt.ls-java-project\bin' 'LinkedList'
Original Linked List:
1 -> 2 -> 4 -> 5 -> 6 -> NULL
Linked List after insertion:
1 -> 2 -> 3 -> 4 -> 5 -> 6 -> NULL
PS C:\Users\HP>
⊗ 0 ⚠ 1   0   Java: Ready
```

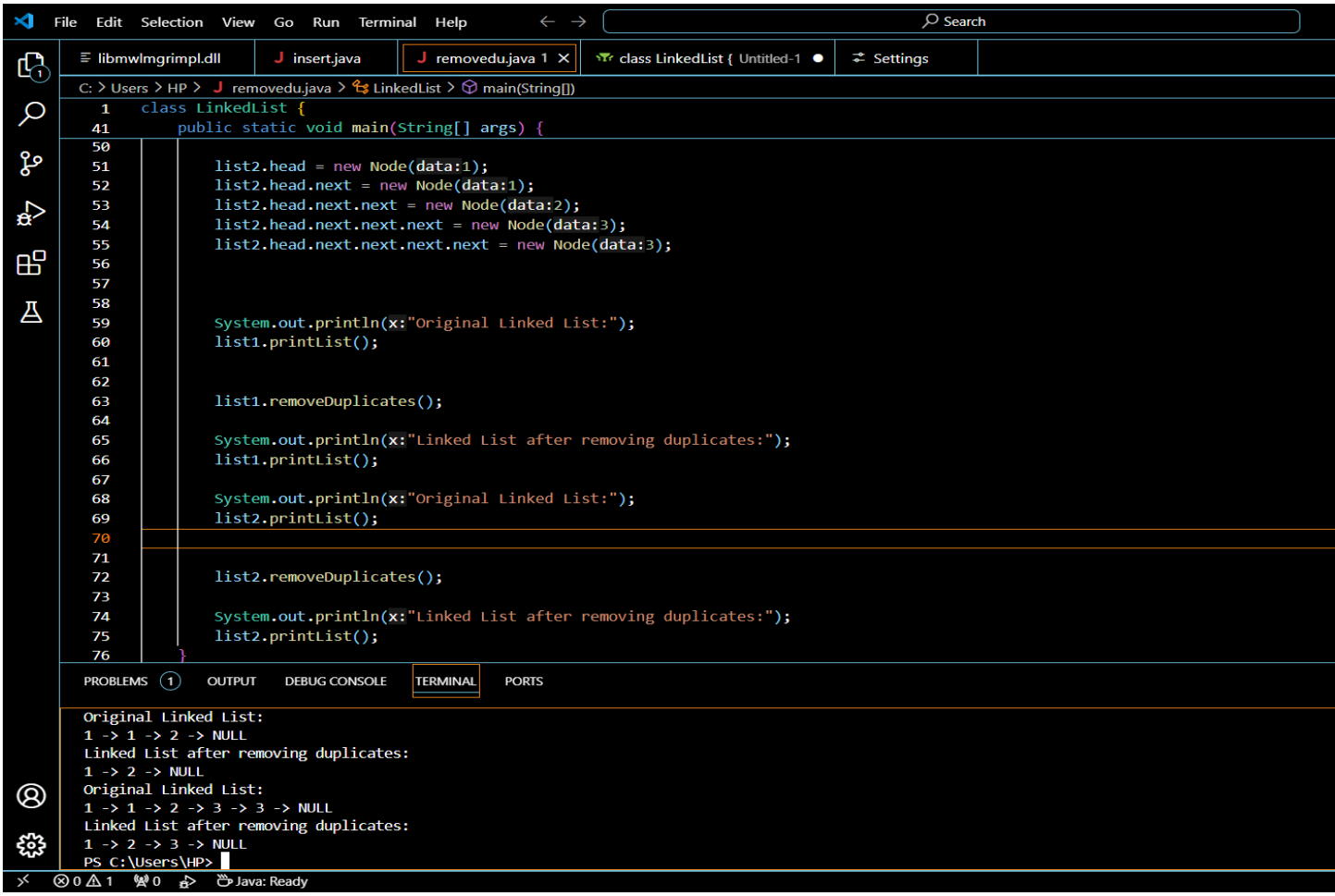
Answer to question no : 3

```
class LinkedList {
    Node head;
    static class Node {
        int data;
        Node next;

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



    public void removeDuplicates() {
        Node current = head;
```

```java
        while (current != null && current.next != null) {
            if (current.data == current.next.data) {

                current.next = current.next.next;
            } else {

                current = current.next;
            }
        }
    }


    public void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " -> ");
            temp = temp.next;
        }
        System.out.println("NULL");
    }

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


        list.head = new Node(1);
        list.head.next = new Node(1);
        list.head.next.next = new Node(2);
        list.head.next.next.next = new Node(3);
        list.head.next.next.next.next = new Node(3);

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


        list.removeDuplicates();

        System.out.println("Linked List after removing duplicates:");
        list.printList();
    }
```

```
}
```

Output :



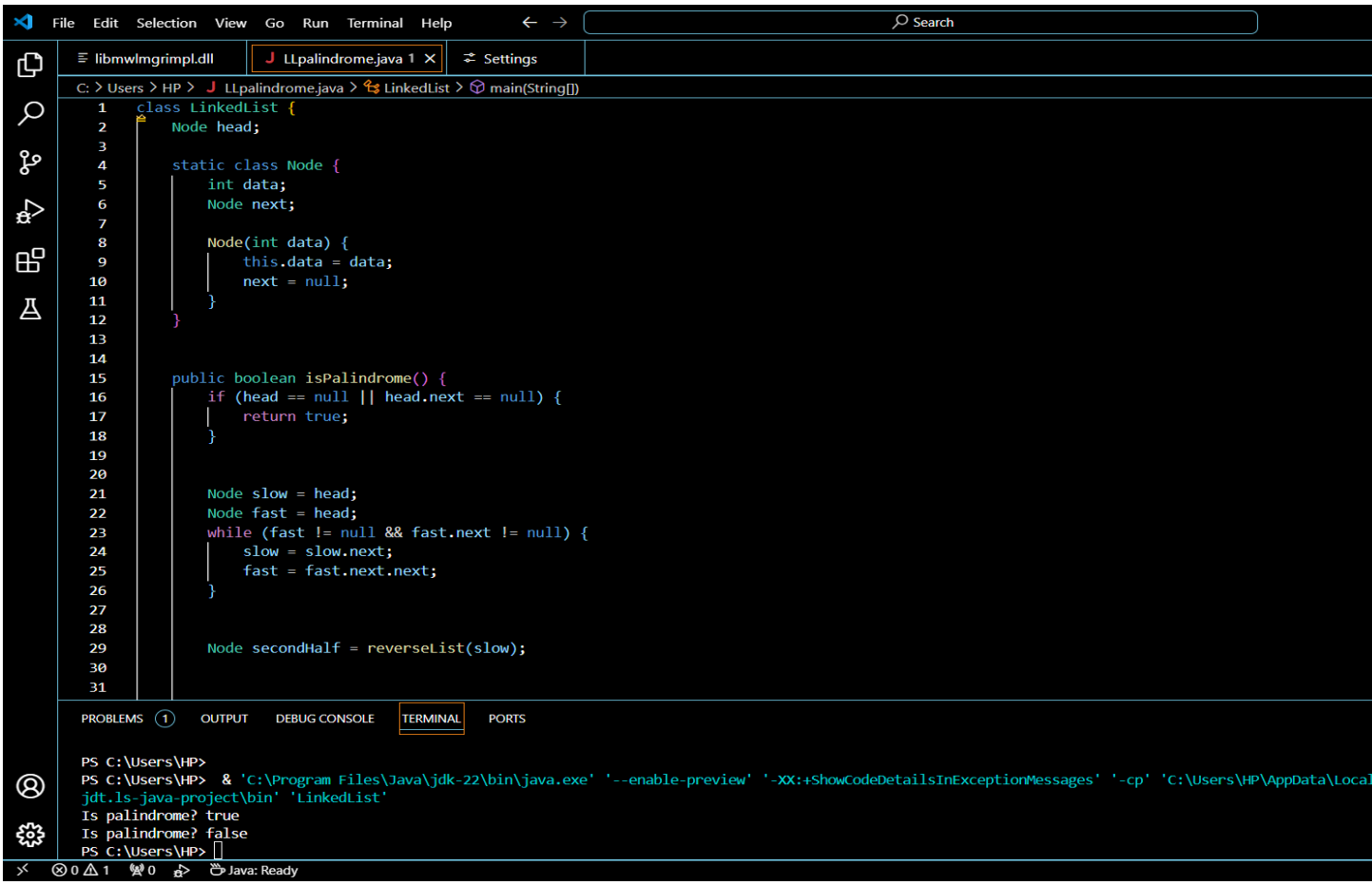
Answer to question no : 4

```
class LinkedList {
    Node head;

    static class Node {
        int data;
        Node next;

        Node(int data) {
```

```
            this.data = data;
            next = null;
        }
    }


    public boolean isPalindrome() {
        if (head == null || head.next == null) {
            return true;
        }


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


        Node secondHalf = reverseList(slow);


        Node firstHalf = head;
        Node secondHalfCopy = secondHalf;
        while (secondHalf != null) {
            if (firstHalf.data != secondHalf.data) {
                return false;
            }
            firstHalf = firstHalf.next;
            secondHalf = secondHalf.next;
        }


        reverseList(secondHalfCopy);

        return true;
    }

    private Node reverseList(Node head) {
        Node prev = null;
        Node current = head;
```

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

    public void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " -> ");
            temp = temp.next;
        }
        System.out.println("NULL");
    }

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


        list.head = new Node(1);
        list.head.next = new Node(2);
        list.head.next.next = new Node(2);
        list.head.next.next.next = new Node(1);

        System.out.println("Is palindrome? " + list.isPalindrome());

        LinkedList list2 = new LinkedList();
        list2.head = new Node(1);
        list2.head.next = new Node(2);

        System.out.println("Is palindrome? " + list2.isPalindrome());
    }
}
```

Output :



Answer to question no : 5

```java
class LinkedList {
    Node head;

    static class Node {
        int data;
        Node next;

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

    public void reverseList() {
        Node prev = null;
        Node current = head;
        while (current != null) {
```

```
            Node nextNode = current.next;
            current.next = prev;
            prev = current;
            current = nextNode;
        }
        head = prev;
    }


    public static LinkedList addTwoNumbers(LinkedList l1, LinkedList l2) {

        l1.reverseList();
        l2.reverseList();


        Node p1 = l1.head;
        Node p2 = l2.head;
        LinkedList result = new LinkedList();
        Node dummyHead = new Node(0);
        Node current = dummyHead;
        int carry = 0;

        while (p1 != null || p2 != null) {
            int x = (p1 != null) ? p1.data : 0;
            int y = (p2 != null) ? p2.data : 0;
            int sum = carry + x + y;
            carry = sum / 10;
            current.next = new Node(sum % 10);
            current = current.next;

            if (p1 != null) p1 = p1.next;
            if (p2 != null) p2 = p2.next;
        }

        if (carry > 0) {
            current.next = new Node(carry);
        }


        result.head = dummyHead.next;
        result.reverseList();
        return result;
    }
```

```java
public void printList() {
    Node temp = head;
    while (temp != null) {
        System.out.print(temp.data + " -> ");
        temp = temp.next;
    }
    System.out.println("NULL");
}

public static void main(String[] args) {
    LinkedList l1 = new LinkedList();
    LinkedList l2 = new LinkedList();
    LinkedList l3 = new LinkedList();
    LinkedList l4 = new LinkedList();




    l1.head = new Node(5);
    l1.head.next = new Node(6);
    l1.head.next.next = new Node(3);



    l2.head = new Node(8);
    l2.head.next = new Node(4);
    l2.head.next.next = new Node(2);

    l3.head = new Node(7);
    l3.head.next = new Node(5);
    l3.head.next.next = new Node(9);
    l3.head.next.next.next = new Node(4);
    l3.head.next.next.next.next = new Node(6);

    l4.head = new Node(8);
    l4.head.next = new Node(4);

    System.out.println("List 1:");
    l1.printList();
    System.out.println("List 2:");
    l2.printList();
    System.out.println("List 3:");
    l3.printList();
```

```java
        System.out.println("List 4:");
        l4.printList();

        LinkedList result = addTwoNumbers(l1, l2);
        System.out.println("Resultant List:");
        result.printList();

        LinkedList result1 = addTwoNumbers(l3, l4);
        System.out.println("Resultant List:");
        result1.printList();



    }
}
```

Output :

```
PS C:\Users\HP>  & 'C:\Program Files\Java\jdk-22\bin\java.exe' '--enable-preview' '-
jdt.ls-java-project\bin' 'LinkedList'
List 1:
5 -> 6 -> 3 -> NULL
List 2:
8 -> 4 -> 2 -> NULL
List 3:
7 -> 5 -> 9 -> 4 -> 6 -> NULL
List 4:
8 -> 4 -> NULL
Resultant List:
1 -> 4 -> 0 -> 5 -> NULL
Resultant List:
7 -> 6 -> 0 -> 3 -> 0 -> NULL
PS C:\Users\HP>
```