Task 2 : You are given a singly linked list. Write a function to find the middle element without using any extra space and only one traversal through the linked list.

```java
package com.wipro.linear;

public class LinkedList {

    private Node head;
    private Node tail;
    private int length;

    class Node {
        int value;
        Node next;

        public Node(int value) {
            super();
            this.value = value;
        }
    }

    public LinkedList(int value) {
        super();
        Node newNode = new Node(value);
        // System.out.println("Node :" +newNode);
```

```java
    public LinkedList(int value) {
        super();
        Node newNode = new Node(value);
        // System.out.println("Node :" +newNode);
        head = newNode;
        tail = newNode;
        length = 1;
    }

    public void getHead() {
        System.out.println("Head :" + head.value);
    }

    public void getTail() {
        System.out.println("Tail : " + tail.value);
    }

    public void getLength() {
        System.out.println("Length  :" + length);
    }
```

```
52
53    public void append(int value) {
54        Node newNode = new Node(value);
55        if (length == 0) {
56
57            head = newNode;
58            tail = newNode;
59        } else {
60            tail.next = newNode;
61            tail = newNode;
62        }
63        length++;
64    }
65
66    public Node findMiddle() {
67        if (length == 0) {
68            return null;
69        }
70        Node slow = head;
71        Node fast = head;
72        while (fast != null && fast.next != null) {
73            slow = slow.next;
```

```
    public Node findMiddle() {
        if (length == 0) {
            return null;
        }
        Node slow = head;
        Node fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        return slow;
    }
```

```
179    public static void main(String[] args) {
180        LinkedList myll = new LinkedList(11);
181        //myll.printList();
182
183        myll.append(3);
184        myll.append(23);
185        myll.append(7);
186        myll.append(45);
187        //myll.printList();
188
189
190
191
192        System.out.println("middle element:"+myll.findMiddle().value);
193
```

Markers | Properties | Servers | Data Source Explorer | Snippets | Terminal | Console

<terminated> LinkedList [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (04-Jun-2024, 9:32:31 pm – 9:32:33 pm) [pid: 6044]

```
middle element:23
```

Task 3: You have a queue of integers that you need to sort. You can only use additional space equivalent to one stack. Describe the steps you would take to sort the elements in the queue.

QueqeSort.java

```java
package com.assignment;

import java.util.*;

public class QueqeSort {
    public static void sortQueue(Queue<Integer> queue) {
        if (queue.size() <= 1) return;

        Queue<Integer> leftHalf = new LinkedList<>();
        Queue<Integer> rightHalf = new LinkedList<>();

        int halfSize = queue.size() / 2;
        for (int i = 0; i < halfSize; i++) {
            leftHalf.add(queue.poll());
        }

        while (!queue.isEmpty()) {
            rightHalf.add(queue.poll());
        }

        sortQueue(leftHalf);
        sortQueue(rightHalf);
        merge(queue, leftHalf, rightHalf);
    }

    private static void merge(Queue<Integer> queue, Queue<Integer> left, Queue<Integer> right) {
        while (!left.isEmpty() && !right.isEmpty()) {
            if (left.peek() <= right.peek()) {
                queue.add(left.poll());
            } else {
                queue.add(right.poll());
            }
        }

        while (!left.isEmpty()) {
            queue.add(left.poll());
        }

        while (!right.isEmpty()) {
            queue.add(right.poll());
        }
    }

    public static void main(String[] args) {
        Queue<Integer> queue = new LinkedList<>();
        queue.add(4);
        queue.add(2);
        queue.add(1);
        queue.add(3);

        sortQueue(queue);
```

```
43
44      public static void main(String[] args) {
45          Queue<Integer> queue = new LinkedList<>();
46          queue.add(4);
47          queue.add(2);
48          queue.add(1);
49          queue.add(3);
50
51          sortQueue(queue);
52
53          // Print sorted queue
54          while (!queue.isEmpty()) {
55              System.out.print(queue.poll() + " ");
56          }
57      }
58  }
```

Markers | Properties | Servers | Data Source Explorer | Snippets | Terminal | Console

```
<terminated> QueqeSort [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe  (05-Jun-2024, 11:18:09 pm – 11:18:10 pm) [pid: 8688]
1 2 3 4
```

Task 4: You must write a function to sort a stack such that the smallest items are on the top. You can use an additional temporary stack, but you may not copy the elements into any other data structure such as an array. The stack supports the following operations: push, pop, peek, and isEmpty .

*StackSort.java | StackSequence.java

```
1  package com.assignment;
2
3  import java.util.Stack;
4
5  public class StackSort {
6      public static void sortStack(Stack<Integer> stack) {
7          Stack<Integer> tempStack = new Stack<>();
8
9          while (!stack.isEmpty()) {
10             int temp = stack.pop();
11             while (!tempStack.isEmpty() && tempStack.peek() > temp) {
12                 stack.push(tempStack.pop());
13             }
14             tempStack.push(temp);
15         }
16
17
18         while (!tempStack.isEmpty()) {
19             stack.push(tempStack.pop());
20         }
21     }
22
23     public static void main(String[] args) {
24         Stack<Integer> stack = new Stack<>();
25         stack.push(4);
26         stack.push(2);
```

*StackSort.java StackSequence.java

```
13              }
14              tempStack.push(temp);
15          }
16
17
18          while (!tempStack.isEmpty()) {
19              stack.push(tempStack.pop());
20          }
21      }
22
23      public static void main(String[] args) {
24          Stack<Integer> stack = new Stack<>();
25          stack.push(4);
26          stack.push(2);
27          stack.push(1);
28          stack.push(3);
29
30          sortStack(stack);
31
32
33          while (!stack.isEmpty()) {
34              System.out.print(stack.pop() + " ");
35          }
36      }
37  }
38
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console

<terminated> StackSort [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (05-Jun-2024, 11:35:45 pm – 11:35:47 pm) [pid: 11384]

```
1 2 3 4
```

Task5: Removing Duplicates from a Sorted Linked List

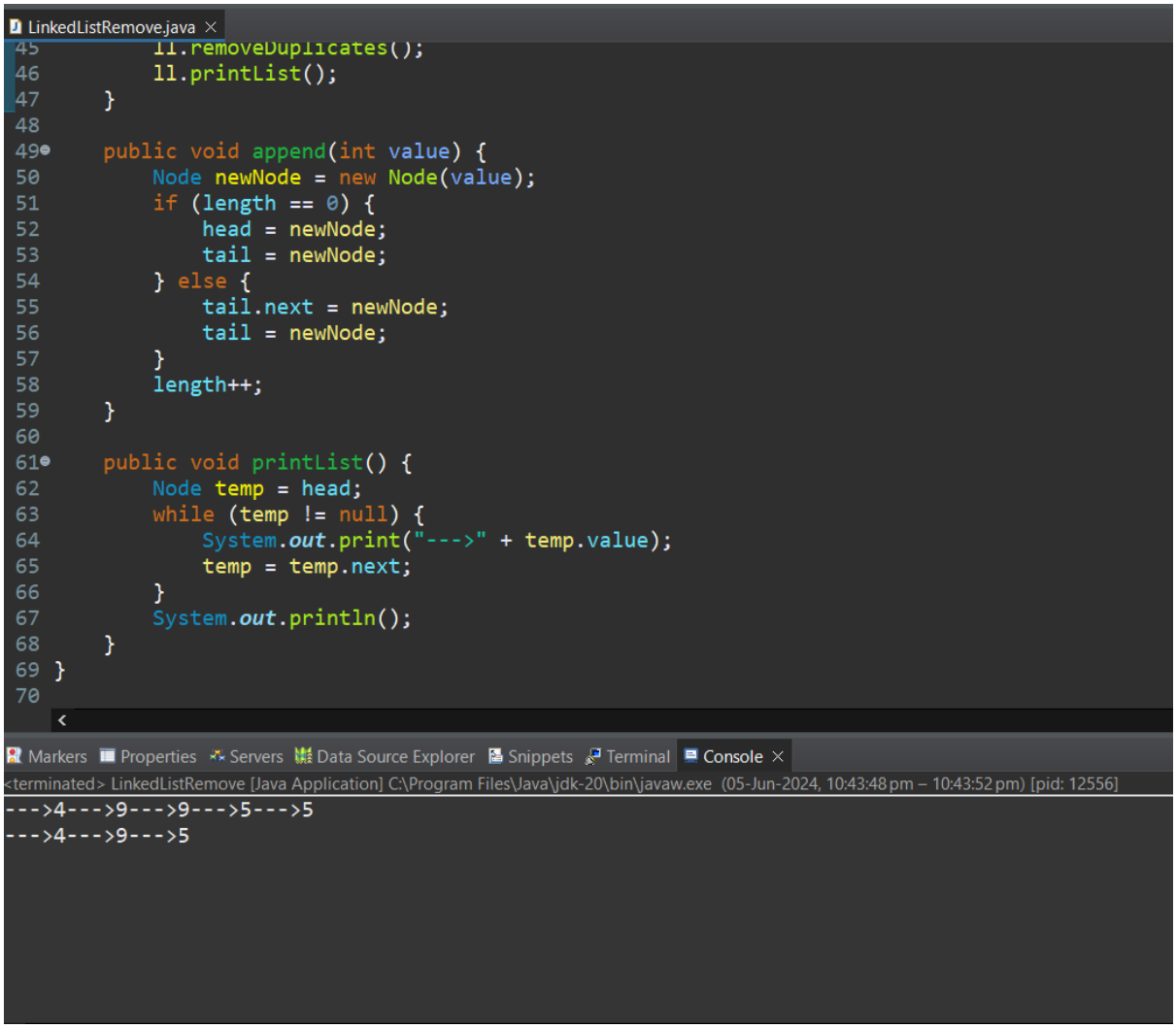
A sorted linked list has been constructed with repeated elements. Describe an algorithm to remove all duplicates from the linked list efficiently

LinkedListRemove.java

```java
package com.assignment;

public class LinkedListRemove {
    private Node tail;
    private Node head;
    private int length;

    class Node {
        int value;
        Node next;

        public Node(int value) {
            super();
            this.value = value;
        }
    }

public LinkedListRemove(int value) {
    super();
   Node newNode = new Node(value);
   head = newNode;
   tail = newNode;
   length = 1;
}

    public void removeDuplicates() {
```

LinkedListRemove.java

```java
    public void removeDuplicates() {
        Node current = head;
        while (current != null && current.next != null) {
            if (current.value == current.next.value) {
                current.next = current.next.next;
                length--;
            } else {
                current = current.next;
            }
        }
    }

    public static void main(String[] args) {
        LinkedListRemove ll = new LinkedListRemove(4);
        ll.append(9);
        ll.append(9);
        ll.append(5);
        ll.append(5);
        ll.printList();
        ll.removeDuplicates();
        ll.printList();
    }

    public void append(int value) {
        Node newNode = new Node(value);
```


LinkedListRemove.java
ll.removeDuplicates();
ll.printList();
}
public void append(int value) {
Node newNode = new Node(value);
if (length == 0) {
head = newNode;
tail = newNode;
} else {
tail.next = newNode;
tail = newNode;
}
length++;
}
public void printList() {
Node temp = head;
while (temp != null) {
System.out.print("--->" + temp.value);
temp = temp.next;
}
System.out.println();
}
}
Markers Properties Servers Data Source Explorer Snippets Terminal Console
<terminated> LinkedListRemove [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (05-Jun-2024, 10:43:48 pm – 10:43:52 pm) [pid: 12556]
--->4--->9--->9--->5--->5
--->4--->9--->5

Task 6:Searching for a Sequence in a Stack

Given a stack and a smaller array representing a sequence, write a function that determines if the sequence is present in the stack. Consider the sequence present if, upon popping the elements, all elements of the array appear consecutively in the stack

```java
package com.assignment;
import java.util.Stack;

public class StackSequence {
    public static boolean isSequenceInStack(Stack<Integer> stack, int[] sequence) {

        if (sequence.length == 0) return true;

        Stack<Integer> tempStack = new Stack<>();
        int sequenceIndex = 0;

        while (!stack.isEmpty()) {
            int currentStackElement = stack.pop();

            if (currentStackElement == sequence[sequenceIndex]) {
                sequenceIndex++;
            } else {

                while (!tempStack.isEmpty()) {
                    stack.push(tempStack.pop());
                }

                sequenceIndex = 0;
            }

```

```java
            if (sequenceIndex == sequence.length) {
                return true;
            }


            tempStack.push(currentStackElement);
        }

        return false;
    }

    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<>();
        stack.push(4);
        stack.push(2);
        stack.push(3);
        stack.push(1);

        int[] sequence = {2, 3};

        boolean sequenceFound = isSequenceInStack(stack, sequence);
        System.out.println("Sequence found in stack: " + sequenceFound);
    }
}
```

Task 7:Merging Two Sorted Linked Lists

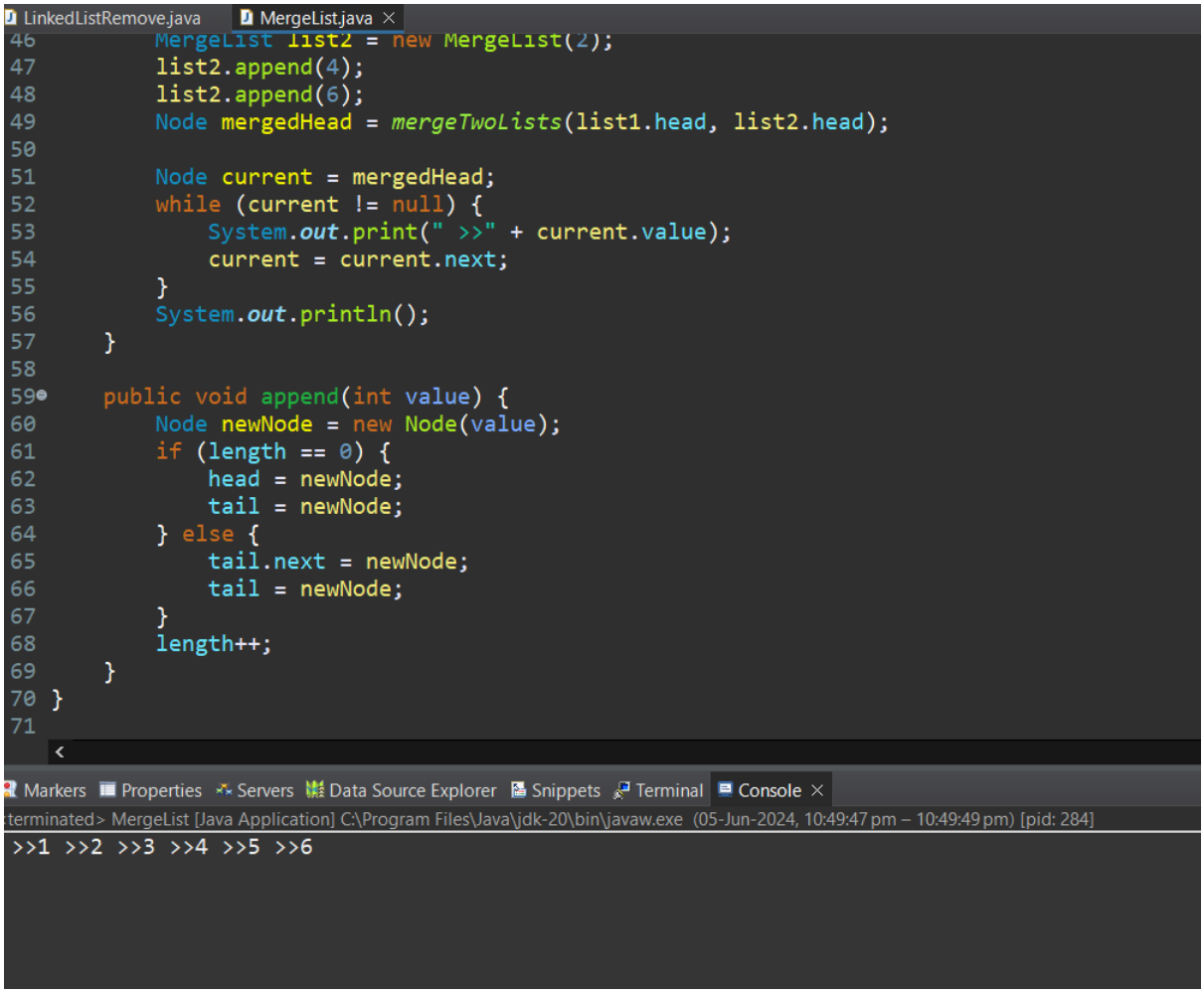
You are provided with the heads of two sorted linked lists. The lists are sorted in ascending order. Create a merged linked list in ascending order from the two input lists without using any extra space (i.e., do not create any new nodes).

LinkedListRemove.java MergeList.java

```java
package com.assignment;

public class MergeList {
    private Node tail;
    private Node head;
    private int length;

    class Node {
        int value;
        Node next;

        public Node(int value) {
            super();
            this.value = value;
        }
    }

    public MergeList(int value) {
        super();
        Node newNode = new Node(value);
        head = newNode;
        tail = newNode;
        length = 1;
    }

    public static Node mergeTwoLists(Node l1, Node l2) {
```

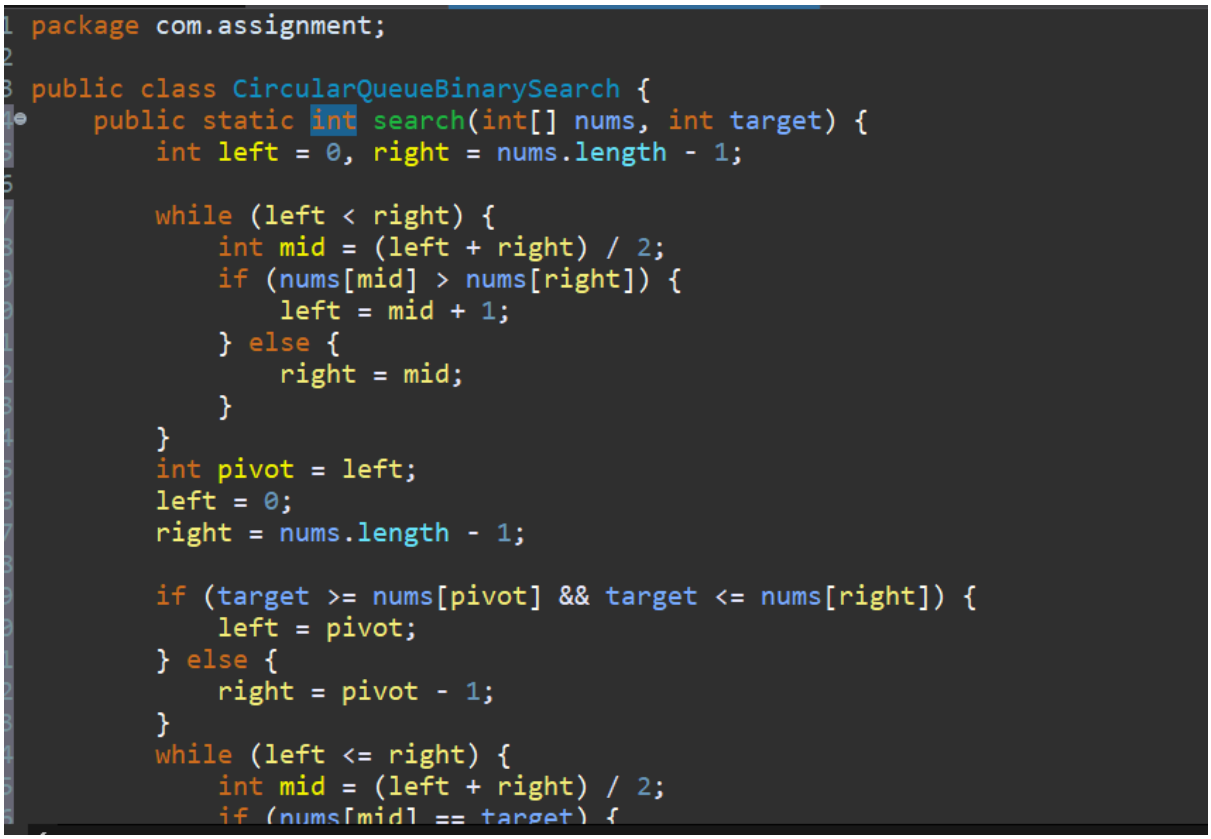
LinkedListRemove.java MergeList.java

```java
    public static Node mergeTwoLists(Node l1, Node l2) {
        if (l1 == null)
            return l2;
        if (l2 == null)
            return l1;

        if (l1.value < l2.value) {
            l1.next = mergeTwoLists(l1.next, l2);
            return l1;
        } else {
            l2.next = mergeTwoLists(l1, l2.next);
            return l2;
        }
    }

    public static void main(String[] args) {
        MergeList list1 = new MergeList(1);
        list1.append(3);
        list1.append(5);

        MergeList list2 = new MergeList(2);
        list2.append(4);
        list2.append(6);
        Node mergedHead = mergeTwoLists(list1.head, list2.head);

```


LinkedListRemove.java
MergeList.java
MergeList list2 = new MergeList(2);
list2.append(4);
list2.append(6);
Node mergedHead = mergeTwoLists(list1.head, list2.head);
Node current = mergedHead;
while (current != null) {
System.out.print(" >>" + current.value);
current = current.next;
}
System.out.println();
}
public void append(int value) {
Node newNode = new Node(value);
if (length == 0) {
head = newNode;
tail = newNode;
} else {
tail.next = newNode;
tail = newNode;
}
length++;
}
}
Markers
Properties
Servers
Data Source Explorer
Snippets
Terminal
Console
terminated> MergeList [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (05-Jun-2024, 10:49:47 pm – 10:49:49 pm) [pid: 284]
>>1 >>2 >>3 >>4 >>5 >>6

Task 8: Circular Queue Binary Search

Consider a circular queue (implemented using a fixedsize array) where the elements are sorted but have been rotated at an unknown index. Describe an approach to perform a binary search for a given element within this circular queue.



```java
package com.assignment;

public class CircularQueueBinarySearch {
    public static int search(int[] nums, int target) {
        int left = 0, right = nums.length - 1;

        while (left < right) {
            int mid = (left + right) / 2;
            if (nums[mid] > nums[right]) {
                left = mid + 1;
            } else {
                right = mid;
            }
        }
        int pivot = left;
        left = 0;
        right = nums.length - 1;

        if (target >= nums[pivot] && target <= nums[right]) {
            left = pivot;
        } else {
            right = pivot - 1;
        }
        while (left <= right) {
            int mid = (left + right) / 2;
            if (nums[mid] == target) {
```

```
        if (target >= nums[pivot] && target <= nums[right]) {
            left = pivot;
        } else {
            right = pivot - 1;
        }
        while (left <= right) {
            int mid = (left + right) / 2;
            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        int[] nums = { 4, 5, 6, 7, 0, 1, 2 };
        int target = 0;
        System.out.println("Index of target: " + search(nums, target));
    }
}
```

Markers | Properties | Servers | Data Source Explorer | Snippets | Terminal | Console

<terminated> CircularQueueBinarySearch [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (05-Jun-2024, 10:53:29 pm – 10:5

```
Index of target: 4
```