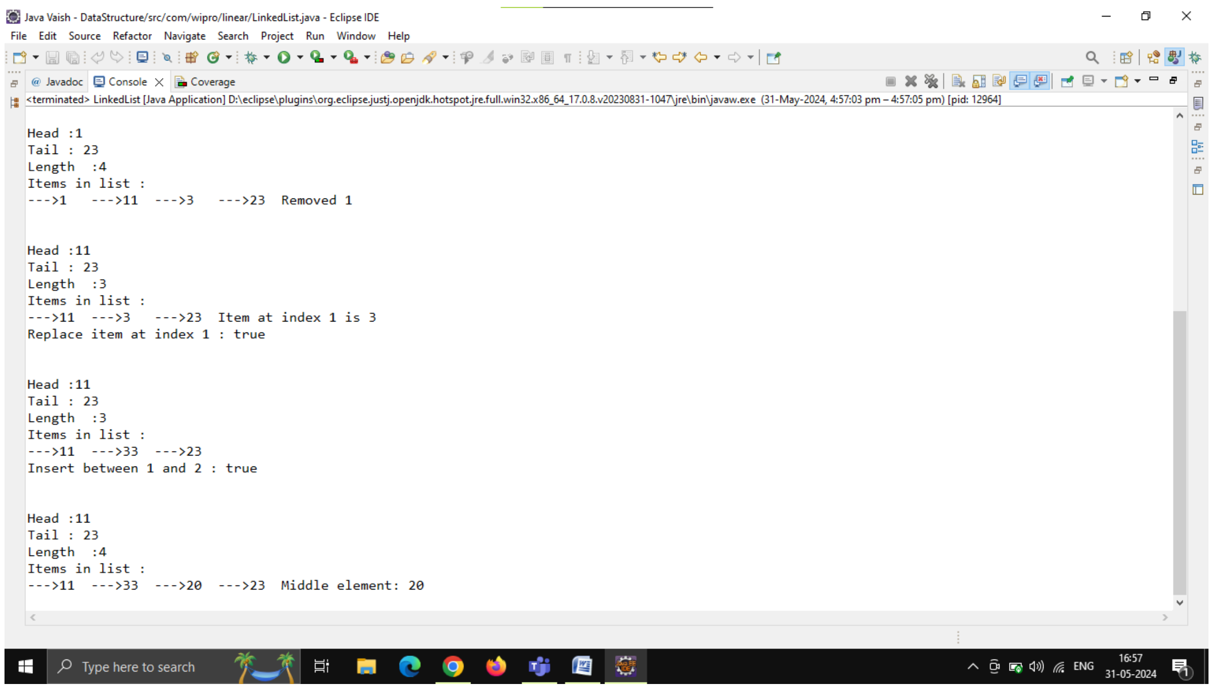
Task : Linked List Middle Element Search

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.

Answer:

```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);
        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);
    }

    public void printList() {
        Node temp = head;
        System.out.println("\n");
        getHead();
        getTail();
        getLength();
        System.out.println("Items in list :");
        while (temp != null) {
            System.out.print("--->" + temp.value + " \t");
            temp = temp.next;
        }
    }

    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 Node removeLast() {
        if (length == 0) {
            return null;
        }

        Node temp = head;
        Node pre = head;
        while (temp.next != null) {
            pre = temp;
            temp = temp.next;
        }

        tail = pre;
        tail.next = null;
        length--;

        if (length == 0) {
            head = null;
            tail = null;
        }
        return temp;
    }

    public void prepend(int value) {
        Node newNode = new Node(value);
        if (length == 0) {
            head = newNode;
            tail = newNode;
        } else {
            newNode.next = head;
            head = newNode;
        }
        length++;
    }

    public Node removeFirst() {
        if (length == 0) {
            return null;
        }
        Node temp = head;
        head = head.next;
        temp.next = null;

        length--;
        if (length == 0) {
            head = null;
            tail = null;
        }

        return temp;
    }

    public Node get(int index) {
        if (index < 0 || index >= length) {
            return null;
        }
        Node temp = head;
        for (int i = 0; i < index; i++) {
            temp = temp.next;
        }

        return temp;
    }

    public boolean set(int index, int value) {
        Node temp = get(index);
        if (temp != null) {
```

```java
            temp.value = value;
            return true;
        }
        return false;
    }

    public boolean insert(int index, int value) {
        if (index < 0 || index >= length) {
            return false;
        }
        if (index == 0) {
            prepend(value);
            return true;
        }
        if (index == length) {
            append(value);
            return true;
        }

        Node newNode = new Node(value);
        Node temp = get(index - 1);
        newNode.next = temp.next;
        temp.next = newNode;
        length++;
        return true;
    }

    public Node findMiddleElement() {
        Node slowPointer = head;
        Node fastPointer = head;

        while (fastPointer != null && fastPointer.next != null) {
            slowPointer = slowPointer.next;
            fastPointer = fastPointer.next.next;
        }

        return slowPointer;
    }

    public static void main(String[] args) {
        LinkedList myll = new LinkedList(11);
        myll.printList();

        myll.append(3);
        myll.append(23);
        myll.append(7);
        myll.printList();
        System.out.println("Removed " + myll.removeLast().value);
        myll.printList();

        myll.prepend(1);
        myll.printList();

        System.out.println("Removed " + myll.removeFirst().value);
        myll.printList();

        System.out.println("Item at index 1 is " + myll.get(1).value);

        System.out.println("Replace item at index 1 : " + myll.set(1,
33));
        myll.printList();

        System.out.println("\nInsert between 1 and 2 : " +
myll.insert(2, 20));
        myll.printList();

        System.out.println("Middle element: " +
myll.findMiddleElement().value);
    }
}
```



Java Vaish - DataStructure/src/com/wipro/linear/LinkedList.java - Eclipse IDE

```
<terminated> LinkedList [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (31-May-2024, 4:57:03 pm – 4:57:05 pm) [pid: 12964]

Head :1
Tail : 23
Length  :4
Items in list :
--->1   --->11  --->3   --->23  Removed 1


Head :11
Tail : 23
Length  :3
Items in list :
--->11  --->3   --->23  Item at index 1 is 3
Replace item at index 1 : true


Head :11
Tail : 23
Length  :3
Items in list :
--->11  --->33  --->23
Insert between 1 and 2 : true


Head :11
Tail : 23
Length  :4
Items in list :
--->11  --->33  --->20  --->23  Middle element: 20
```

Task 3: Queue Sorting with Limited Space

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

Answer:

1. **Enqueue all elements into the stack**: Dequeue each element from the queue and enqueue it into the stack. This will reverse the order of elements in the stack compared to the original queue.
2. **Sort the stack**: Implement a sorting algorithm, such as bubble sort, insertion sort, or merge sort, using the stack as the data structure. You can use the queue to simulate the stack operations as needed.
3. **Dequeue elements back into the queue**: After sorting the stack, dequeue each element from the stack and enqueue it back into the queue. Since elements are dequeued in sorted order from the stack, they will be enqueued in sorted order into the queue.

Here's a more detailed explanation:

plaintext
Copy code
Input: Queue [4, 2, 5, 1, 3]
Output: Sorted Queue [1, 2, 3, 4, 5]

Step 1: Enqueue all elements into the stack
Queue: [4, 2, 5, 1, 3]
Stack: []

Dequeuing elements from the queue and enqueuing them into the stack:
Queue: []
Stack: [4, 2, 5, 1, 3]

Step 2: Sort the stack
Apply any sorting algorithm on the stack. For simplicity, let's use bubble sort:

Stack: [4, 2, 5, 1, 3]

Bubble Sort:
Pass 1: [2, 4, 1, 3, 5]
Pass 2: [2, 1, 3, 4, 5]

Pass 3: [1, 2, 3, 4, 5]

Stack (sorted): [1, 2, 3, 4, 5]

Step 3: Dequeue elements back into the queue
Queue: []
Stack: [1, 2, 3, 4, 5]

Dequeuing elements from the stack and enqueuing them into the queue:
Queue: [1, 2, 3, 4, 5]
Stack: []

Sorted Queue: [1, 2, 3, 4, 5]

This approach sorts the elements in the queue using only one additional stack and maintains a space complexity equivalent to the size of one stack.

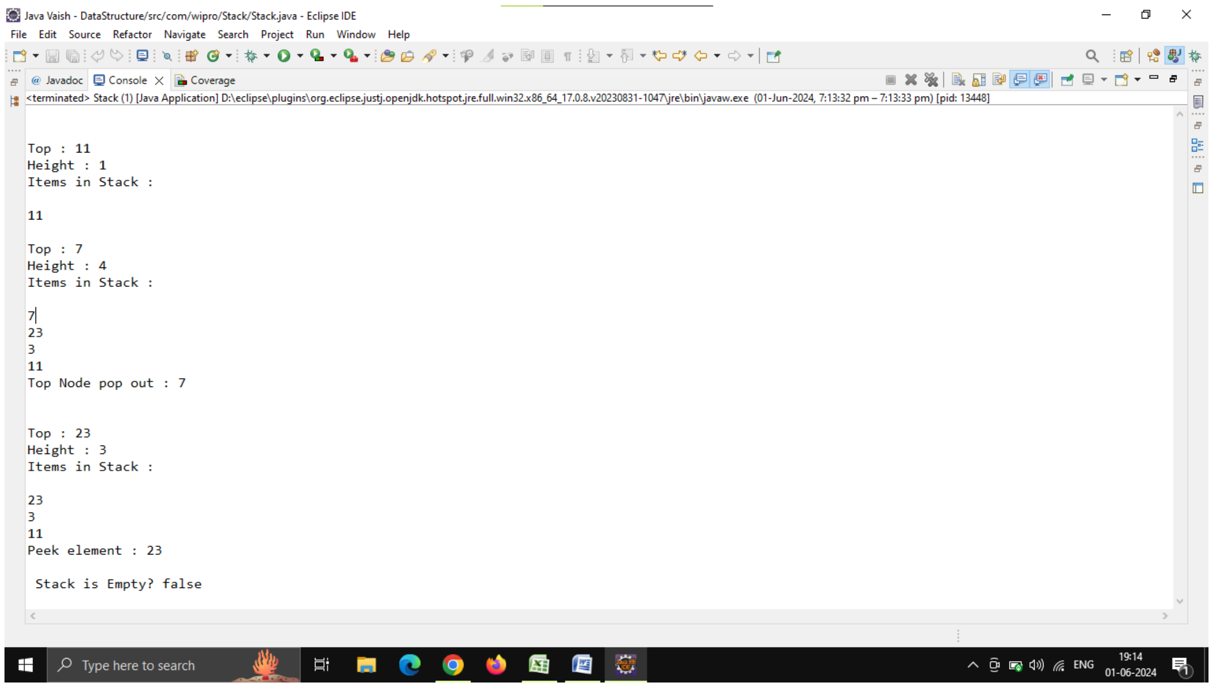
Task 4: Stack Sorting In-Place

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.

Answer:

```java
package com.wipro.Stack;

public class Stack {

	private Node top;
	private int height;

	class Node {
		int value;
		Node next;
```

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

    public Stack(int value) {
        Node newNode = new Node(value);
        top = newNode;
        height = 1;
    }

    public void getHeight() {
        System.out.println("Height : " + height);
    }

    public void getTop() {
        System.out.println("Top : " + top.value);
    }

    public void printStack() {
        Node temp = top;
        System.out.println("\n");
        getTop();
        getHeight();

        System.out.println("Items in Stack :");
        while (temp != null) {
            System.out.print("\n" + temp.value);
            temp = temp.next;
        }
    }

    public void push(int value) {
        Node newNode = new Node(value);
        if (height == 0) {
            top = newNode;
        } else {
            newNode.next = top;
            top = newNode;
        }
        height++;
    }

    public Node pop() {
        if (height == 0) {
                return null;
            }
            Node temp = top;
            top = top.next;
            temp.next = null;
            height--;

            return temp;

      }

      public int peek() {
            if(top == null) {
                  System.out.println("Stack is empty");
                  return -1;
            }
            return top.value;
      }

      public boolean isEmpty() {
            return height == 0;
      }

      public static void main(String[] args) {
            Stack mystack = new Stack(11);
            // mystack.getHeight();
            // mystack.getTop();
            mystack.printStack();

            mystack.push(3);
            mystack.push(23);
            mystack.push(7);
            mystack.printStack();

            System.out.println("\nTop Node pop out : "
+mystack.pop().value);
            mystack.printStack();

            System.out.println("\nPeek element : " + mystack.peek());

            System.out.println("\n Stack is Empty? "+
mystack.isEmpty());
      }

}
```


Java Vaish - DataStructure/src/com/wipro/Stack/Stack.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Javadoc Console Coverage
Type here to search
ENG
19:14
01-06-2024

```
<terminated> Stack (1) [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (01-Jun-2024, 7:13:32 pm – 7:13:33 pm) [pid: 13448]


Top : 11
Height : 1
Items in Stack :

11

Top : 7
Height : 4
Items in Stack :

7
23
3
11
Top Node pop out : 7


Top : 23
Height : 3
Items in Stack :

23
3
11
Peek element : 23

 Stack is Empty? false
```

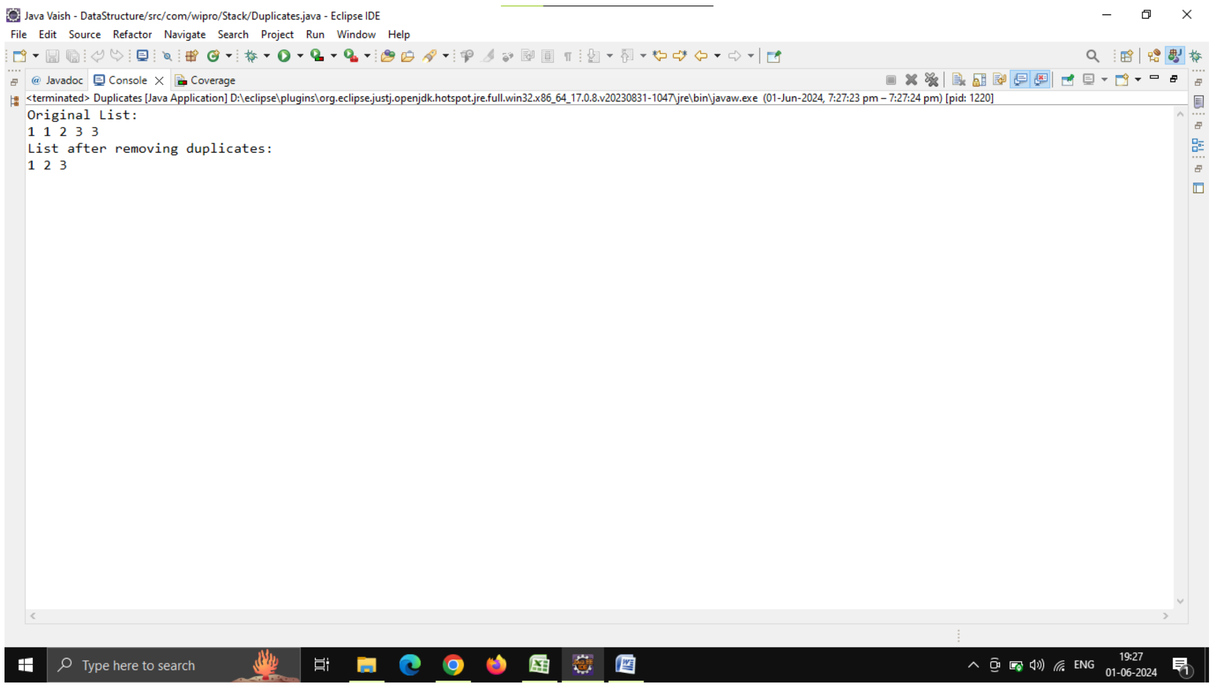
Task 5: Removing Duplicates from a Sorted Linked List

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

<u>Answer:</u>

```java
package com.wipro.Stack;

class ListNode {
     int val;
     ListNode next;
     ListNode(int x)
     {
       val = x;
       }
}

public class Duplicates {
     public ListNode deleteDuplicates(ListNode head) {
          if (head == null) return null;

          ListNode current = head;
```

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

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

                current = current.next;
            }
        }

        return head;
    }


    public void printList(ListNode head) {
        ListNode current = head;
        while (current != null) {
            System.out.print(current.val + " ");
            current = current.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
       Duplicates remover = new Duplicates();

        ListNode head = new ListNode(1);
        head.next = new ListNode(1);
        head.next.next = new ListNode(2);
        head.next.next.next = new ListNode(3);
        head.next.next.next.next = new ListNode(3);

        System.out.println("Original List:");
        remover.printList(head);

        head = remover.deleteDuplicates(head);

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


Java Vaish - DataStructure/src/com/wipro/Stack/Duplicates.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Javadoc
Console
Coverage
Type here to search
ENG
19:27
01-06-2024

```
<terminated> Duplicates [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (01-Jun-2024, 7:27:23 pm – 7:27:24 pm) [pid: 1220]
Original List:
1 1 2 3 3
List after removing duplicates:
1 2 3
```

Explanation of the Code:

ListNode class: A simple class to define the structure of a linked list node.

deleteDuplicates method: This method removes duplicates from the sorted linked list.

It first checks if the head is null, in which case it returns null.

It then initializes the current pointer to the head.

In the while loop, it compares the current node's value with the next node's value. If they are equal, it skips the next node by changing the next reference of the current node. Otherwise, it moves the current pointer to the next node.

printList method: A helper method to print the elements of the linked list, useful for testing.

main method: Sets up a test case, prints the original list, removes duplicates, and then prints the modified list.

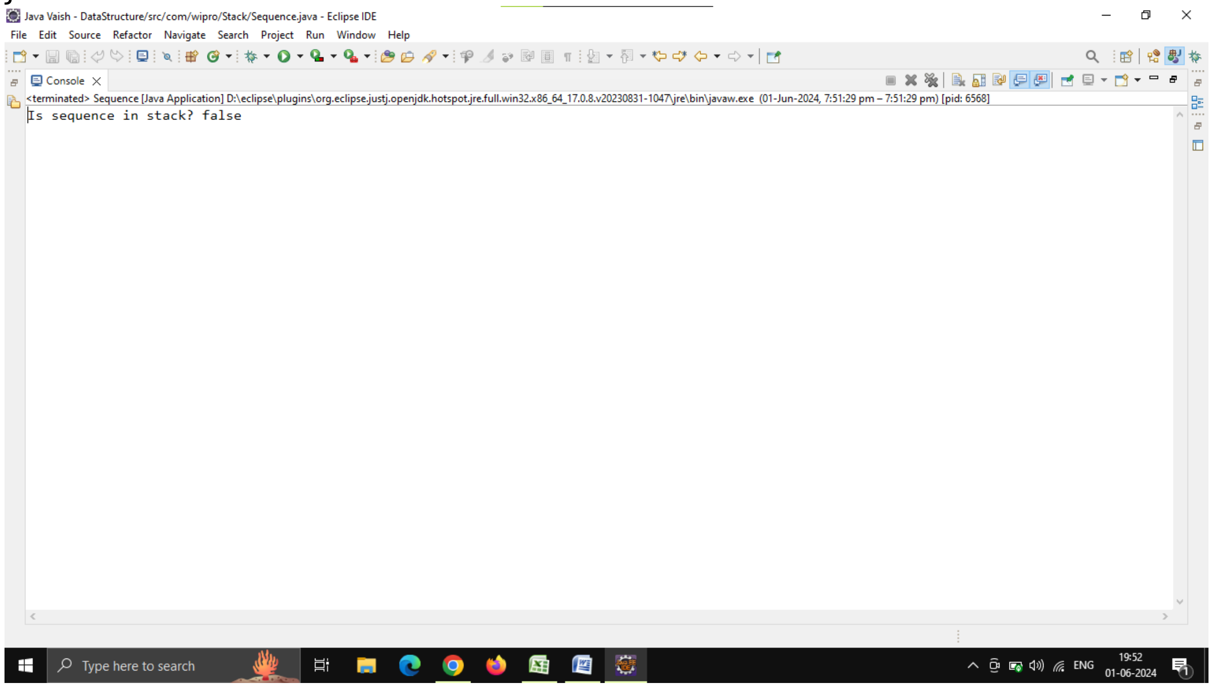
This algorithm runs in O(n) time complexity where n is the number of nodes in the linked list, making it efficient for removing duplicates from a sorted linked list.

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, upion popping the elements, all elements of the array appear consecutively in the stack

Answer:

```java
package com.wipro.Stack;

import java.util.Stack;

public class Sequence{

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

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

        if (stack.isEmpty()) {
            return false;
        }

        Stack<Integer> tempStack = new Stack<>();

        int seqIndex = 0;

        while (!stack.isEmpty()) {
            int poppedElement = stack.pop();
            tempStack.push(poppedElement);

            if (poppedElement == sequence[seqIndex]) {
                seqIndex++;

                if (seqIndex == sequence.length) {
                    return true;
                }
            } else {

                seqIndex = 0;
            }
        }
        return false;
    }

    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<>();
        stack.push(1);
        stack.push(2);
        stack.push(3);
        stack.push(4);
        stack.push(5);

        int[] sequence = {3, 4, 5};

        boolean result = isSequenceInStack(stack, sequence);
        System.out.println("Is sequence in stack? " + result);
    }
}
```


Java Vaish - DataStructure/src/com/wipro/Stack/Sequence.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console
Type here to search
ENG
19:52
01-06-2024

```
<terminated> Sequence [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (01-Jun-2024, 7:51:29 pm – 7:51:29 pm) [pid: 6568]
Is sequence in stack? false
```

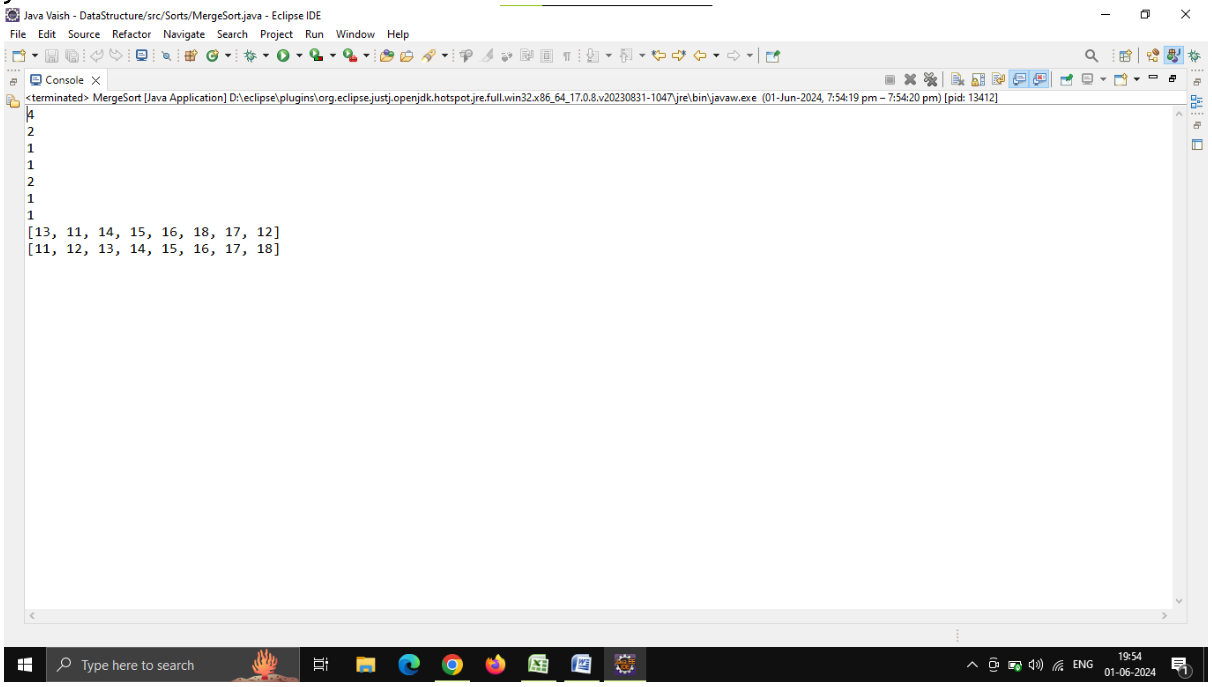
Task 7: Merging Two Sorted Linked Lists

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) e., do not create any new nodes)

Answer:

```
package Sorts;
import java.util.Arrays;

public class MergeSort {

      public static void main(String[] args) {
            int[] arr = { 13, 11, 14, 15, 16, 18, 17, 12 };

            int[] sortedArray = mergeSort(arr);

            System.out.println(Arrays.toString(arr));
            System.out.println(Arrays.toString(sortedArray));

      }

      private static int[] mergeSort(int[] arr) {

            if (arr.length == 1) {
                  return arr;
            }

            int midIndex = arr.length / 2;
            System.out.println(midIndex);

            int[] left = mergeSort(Arrays.copyOfRange(arr, 0,
midIndex));

            int[] right = mergeSort(Arrays.copyOfRange(arr, midIndex,
arr.length));

            return merge(left, right);
      }

      private static int[] merge(int[] left, int[] right) {
            int[] combined = new int[left.length + right.length];
            int index = 0;
            int i = 0;
            int j = 0;

            while (i < left.length && j < right.length) {
                  if (left[i] < right[j]) {
                        combined[index] = left[i];
                        index++;
                        i++;
                  } else {
```

```
                combined[index] = right[j];
                index++;
                j++;
            }

        }
        while (i < left.length) {
            combined[index] = left[i];
            index++;
            i++;
        }
        while (j < right.length) {
            combined[index] = right[j];
            index++;
            j++;
        }

        return combined;
    }

}
```


Java Vaish - DataStructure/src/Sorts/MergeSort.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console
Type here to search
ENG
19:54
01-06-2024

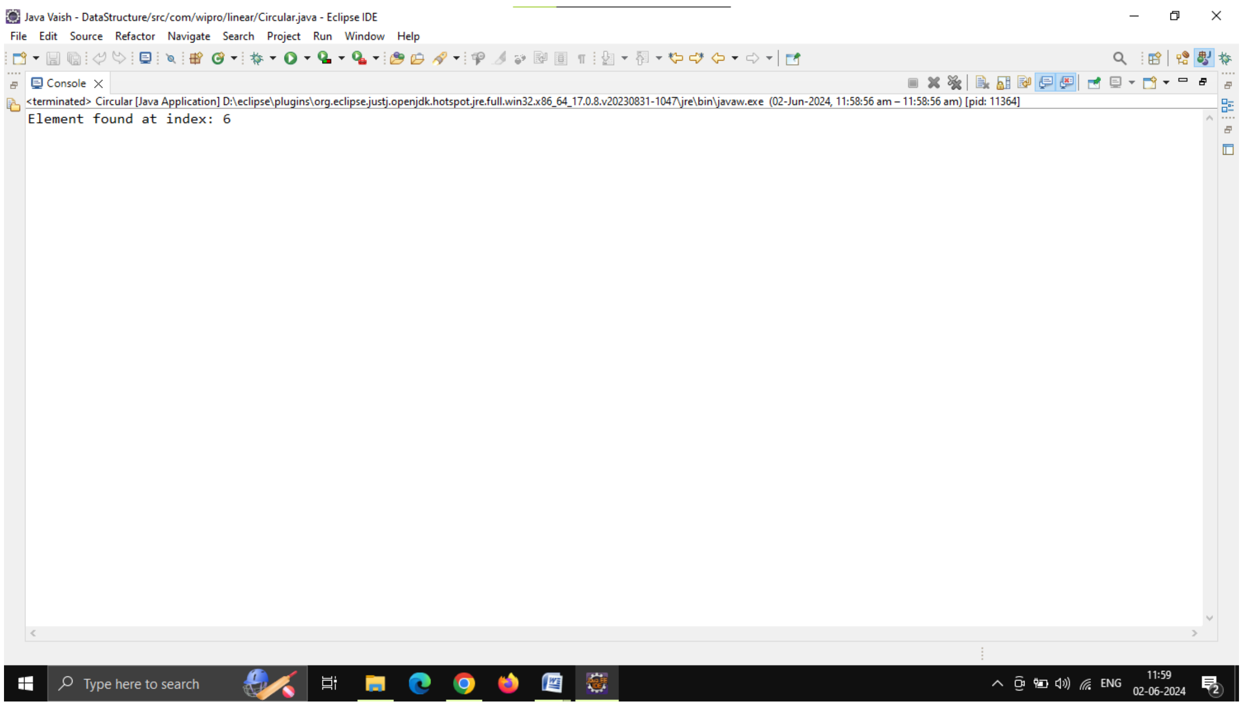
```
<terminated> MergeSort [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (01-Jun-2024, 7:54:19 pm – 7:54:20 pm) [pid: 13412]
4
2
1
1
2
1
1
[13, 11, 14, 15, 16, 18, 17, 12]
[11, 12, 13, 14, 15, 16, 17, 18]
```

Task 8: Circular Queue Binary Search
Consider a circular queue (implemented using a fixed-size 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.

Answer:

```
package com.wipro.linear;

public class Circular {

    public static int circularBinarySearch(int[] array, int target) {
        int rotationPoint = findRotationPoint(array);
        int n = array.length;
        if (target >= array[0] && target <= array[n - 1]) {
            return binarySearch(array, target, 0, n - 1);
        } else if (target >= array[0]) {
            return binarySearch(array, target, 0, rotationPoint - 1);
        } else {
            return binarySearch(array, target, rotationPoint, n - 1);
        }
    }

    private static int binarySearch(int[] array, int target, int low,
int high) {
        while (low <= high) {
            int mid = (low + high) / 2;
            int adjustedMid = mid % array.length;
            if (array[adjustedMid] == target) {
                return adjustedMid;
            } else if (array[adjustedMid] < target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return -1;
    }

    private static int findRotationPoint(int[] array) {
        int low = 0, high = array.length - 1;
        while (low < high) {
```

```java
            int mid = (low + high) / 2;
            if (array[mid] > array[high]) {
                low = mid + 1;
            } else {
                high = mid;
            }
        }
        return low;
    }

    public static void main(String[] args) {
        int[] array = {6, 7, 8, 9, 1, 2, 3, 4, 5};
        int target = 3;
        int index = circularBinarySearch(array, target);
        if (index != -1) {
            System.out.println("Element found at index: " + index);
        } else {
            System.out.println("Element not found");
        }
    }
}
```



Java Vaish - DataStructure/src/com/wipro/linear/Circular.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console
Type here to search
ENG
11:59
02-06-2024

```
<terminated> Circular [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (02-Jun-2024, 11:58:56 am – 11:58:56 am) [pid: 11364]
Element found at index: 6
```