

Lesson 02 Demo 09

Merge Two Sorted Linked Lists

Objective: To write a function that merges two sorted linked lists into a single sorted linked list, and the merged list should be made by splicing together the nodes of the first two lists

Tools required: Visual Studio Code (VS Code) and JavaScript

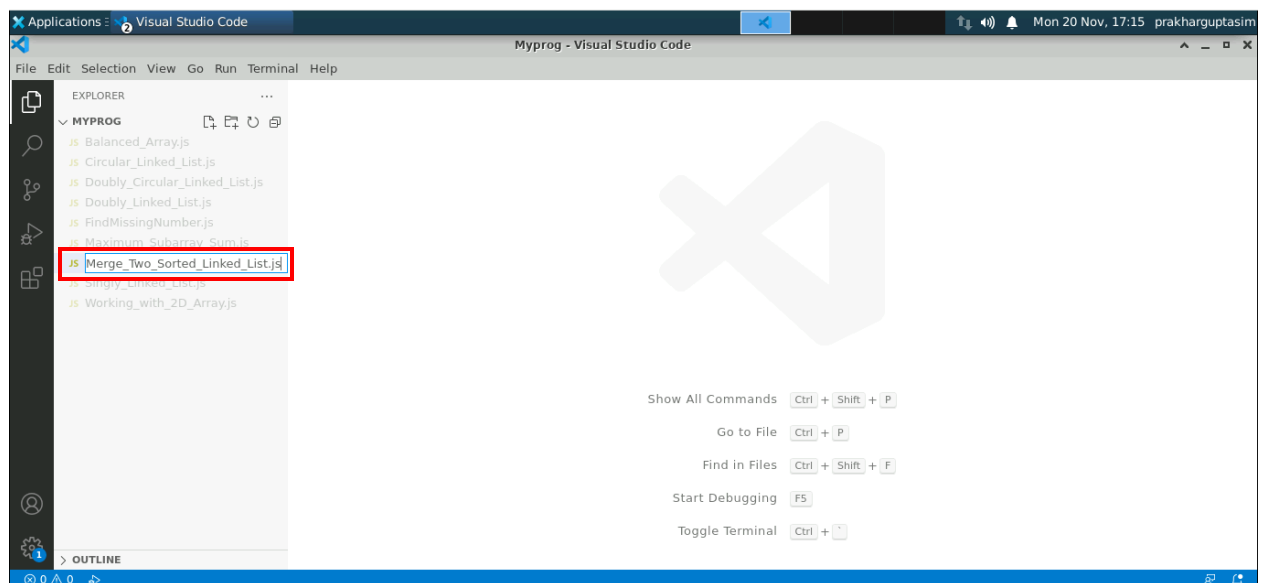
Prerequisites: Perform demo 01 of lesson 02

Steps to be followed:

1. Create and execute the JS file

Step 1: Create and execute the JS file

1.1 Create a JavaScript file named **Merge_Two_Sorted_Linked_List.js** as shown below:



1.2 Write the code in the file created in step 1.1 as shown below:

```
class ListNode {
  constructor(value) {
    this.value = value;
    this.next = null;
  }
}

function mergeSortedLists(l1, l2) {
  let dummyHead = new ListNode(0);
  let current = dummyHead;

  while (l1 !== null && l2 !== null) {
    if (l1.value < l2.value) {
      current.next = l1;
      l1 = l1.next;
    } else {
      current.next = l2;
      l2 = l2.next;
    }
    current = current.next;
  }

  // At least one of l1 and l2 can still have nodes at this point, so connect
  // the non-null list to the end of the merged list.
  current.next = l1 === null ? l2 : l1;

  return dummyHead.next;
}
```

```

1 > class ListNode {
2   value;
3   next;
4 }
5
6
7 function mergeSortedLists(l1, l2) {
8   let dummyHead = new ListNode(0);
9   let current = dummyHead;
10
11   while (l1 !== null && l2 !== null) {
12     if (l1.value < l2.value) {
13       current.next = l1;
14       l1 = l1.next;
15     } else {
16       current.next = l2;
17       l2 = l2.next;
18     }
19     current = current.next;
20   }
21
22   // At least one of l1 and l2 can still have nodes at this point, so connect
23   // the non-null list to the end of the merged list.
24   current.next = l1 === null ? l2 : l1;
25
26   return dummyHead.next;
27 }
28

```

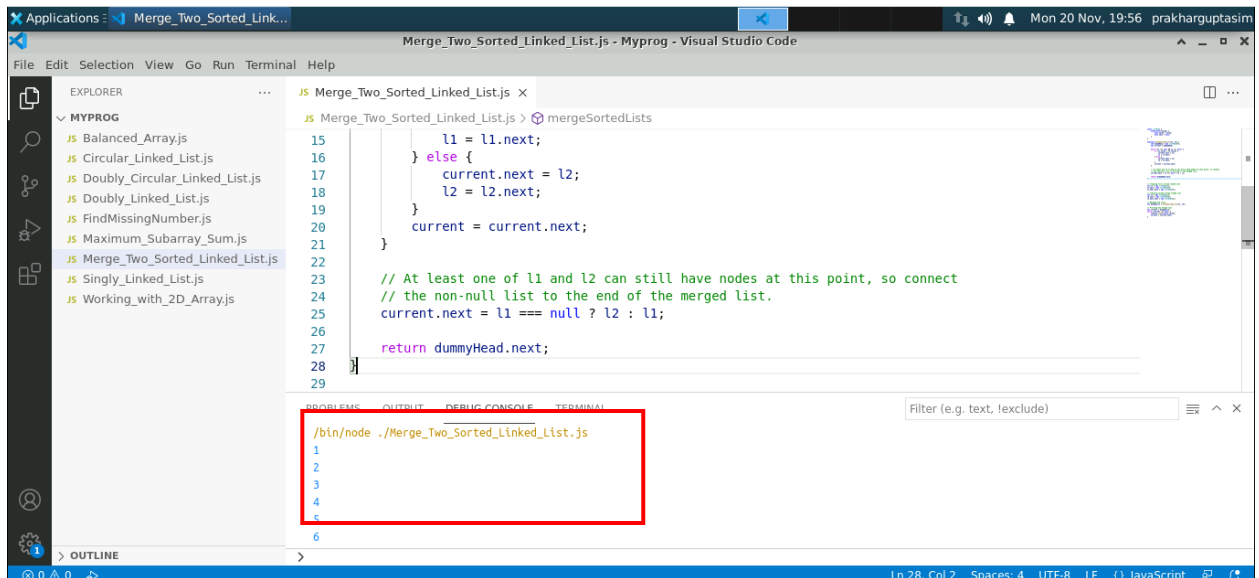
1.3 Save the code and click on **Run->Run Without Debugging** to check the output in the debug console

```

26   }
27   return dummyHead.next;
28 }

```

- Now you can see the output in the debug console as shown below:



```
15     l1 = l1.next;
16   } else {
17     current.next = l2;
18     l2 = l2.next;
19   }
20   current = current.next;
21 }
22
23 // At least one of l1 and l2 can still have nodes at this point, so connect
24 // the non-null list to the end of the merged list.
25 current.next = l1 === null ? l2 : l1;
26
27 return dummyHead.next;
28
29
```

```
/bin/node ./Merge_Two_Sorted_Linked_List.js
1
2
3
4
5
6
```

Explanation:

1. **Initialization:** Create a dummy head node to simplify edge cases and use a current pointer for traversal
2. **Iterative Comparison:** Loop through both lists, comparing the current nodes of each
3. Attach the smaller node to the end of the new list and advance in that list
4. **Appending Remaining Nodes:** If one list is exhausted before the other, directly attach the remaining part of the non-empty list to the merged list
5. **Return Result:** Skip the dummy head and return the next node, which is the start of the merged, sorted list

By following the above steps, you have efficiently combined two sorted linked lists into a single sorted list. This implementation is concise and leverages the existing order of the lists to minimize operations.