Lesson 02 Demo 09

Merging Two Sorted Linked Lists

Objective: To write a function that merges two sorted linked lists into a single sorted linked list by splicing together their nodes, which helps understand merging techniques used in algorithms for efficient list manipulation

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

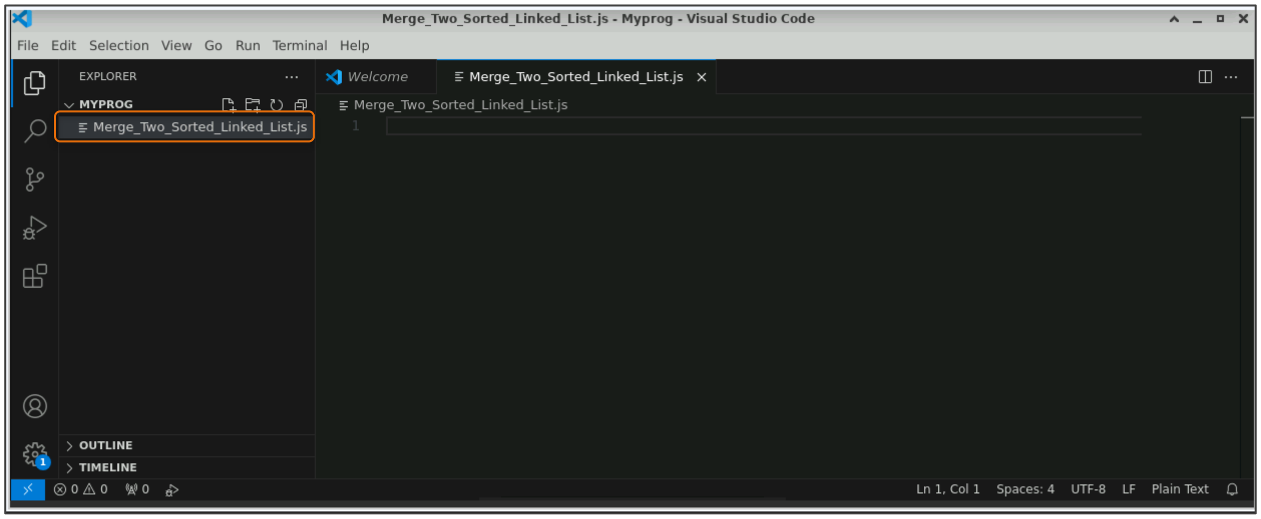
Prerequisites: Completion of Lesson 02 Demo 01

Steps to be followed:

1. Create a JavaScript file and execute it

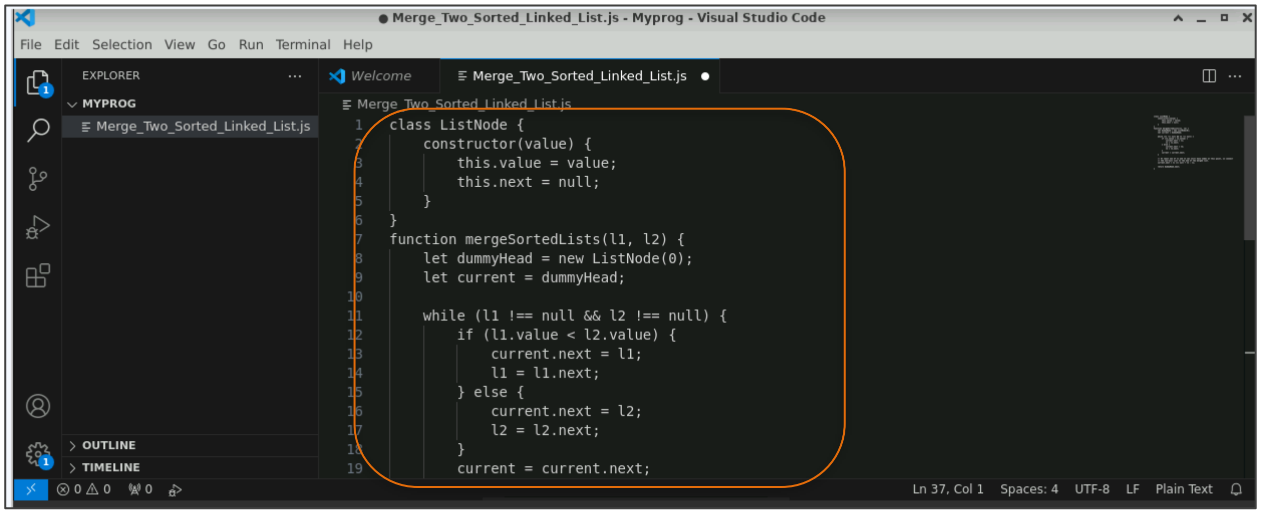
Step 1: Create a JavaScript file and execute it

1.1 Open the Visual Studio Code editor and create a JavaScript file named **Merge_Two_Sorted_Linked_List.js**

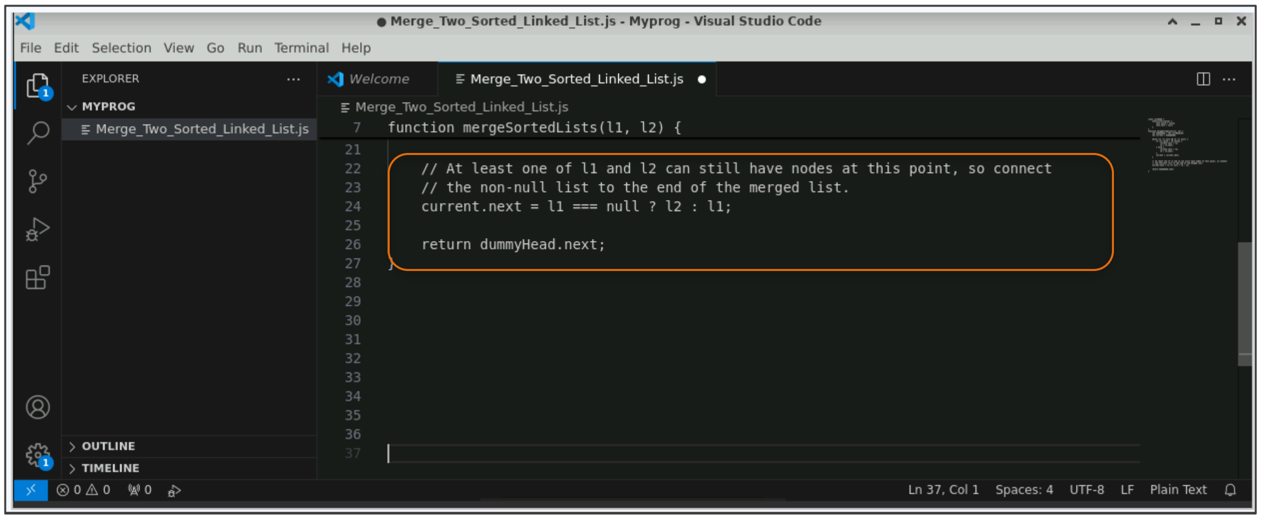

Merge_Two_Sorted_Linked_List.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MYPROG
Merge_Two_Sorted_Linked_List.js
Welcome
Merge_Two_Sorted_Linked_List.js
OUTLINE
TIMELINE
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text

1.2 Add the following code to the file:

```
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;
}
```

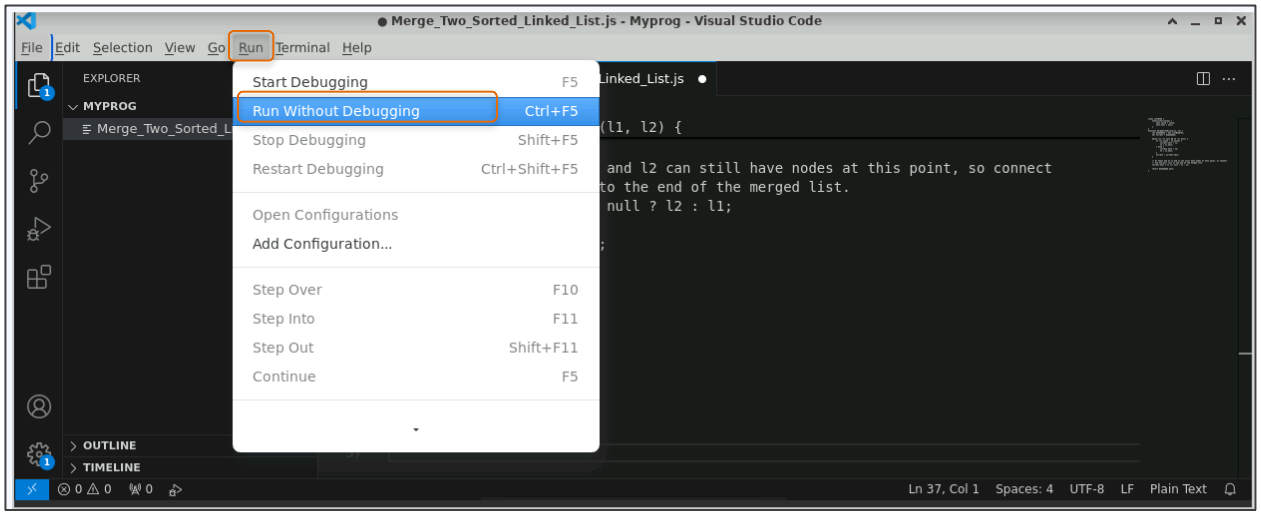


```
class ListNode {
    constructor(value) {
        this.value = value;
        this.next = null;
    }
}
function mergeSortedLists(l1, l2) {
    let dummyHead = new ListNode(0);
    let current = dummyHead;

    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;
```

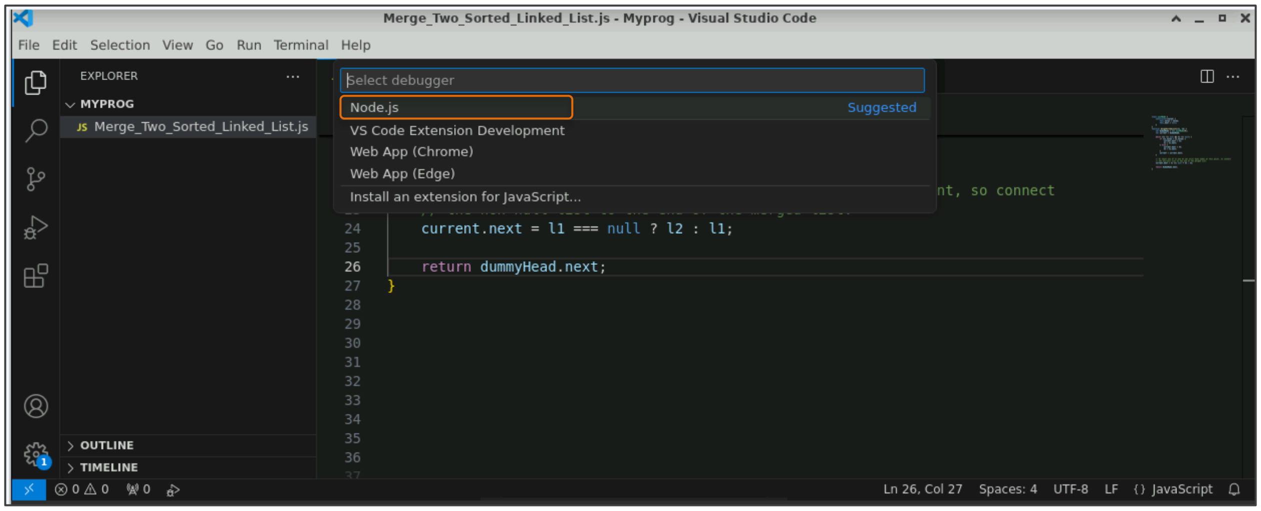


```
function mergeSortedLists(l1, l2) {

    // 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;
}
```

1.3 Click **Run** and then **Run Without Debugging**. Select **Node.js** to check the output in the DEBUG CONSOLE.

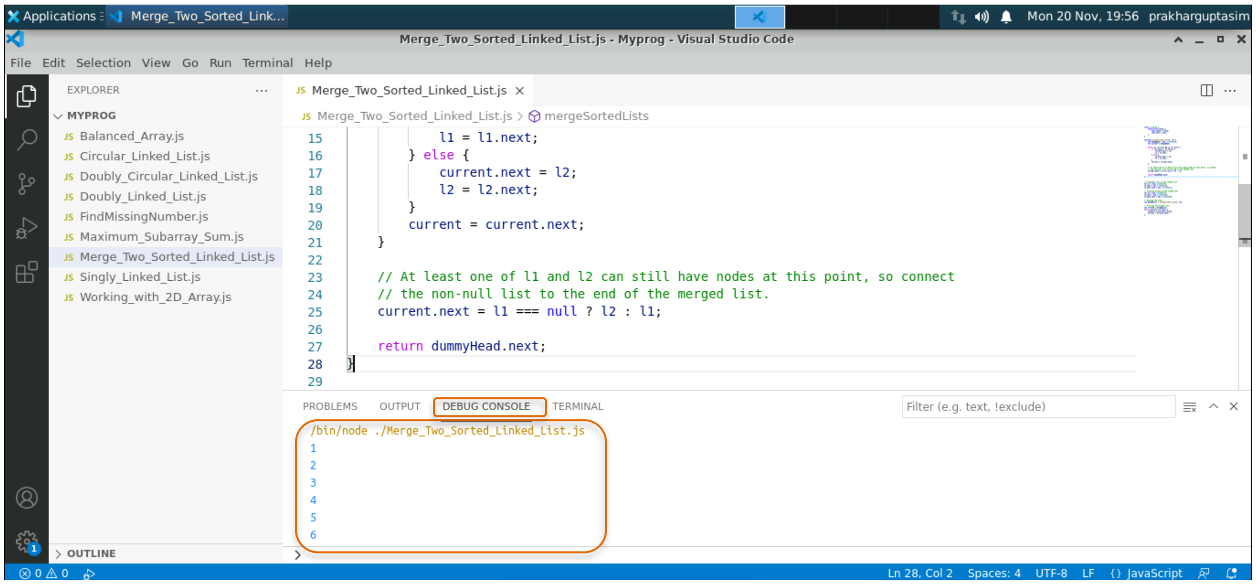

● Merge_Two_Sorted_Linked_List.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MYPROG
Merge_Two_Sorted_L
Start Debugging F5
Run Without Debugging Ctrl+F5
Stop Debugging Shift+F5
Restart Debugging Ctrl+Shift+F5
Open Configurations
Add Configuration...
Step Over F10
Step Into F11
Step Out Shift+F11
Continue F5
OUTLINE
TIMELINE
Ln 37, Col 1 Spaces: 4 UTF-8 LF Plain Text

```
(l1, l2) {

 and l2 can still have nodes at this point, so connect
to the end of the merged list.
 null ? l2 : l1;
```


Merge_Two_Sorted_Linked_List.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MYPROG
Merge_Two_Sorted_Linked_List.js
Select debugger
Node.js
Suggested
VS Code Extension Development
Web App (Chrome)
Web App (Edge)
Install an extension for JavaScript...
OUTLINE
TIMELINE
Ln 26, Col 27 Spaces: 4 UTF-8 LF {} JavaScript

```
nt, so connect
    current.next = l1 === null ? l2 : l1;

    return dummyHead.next;
}
```

1.4 View the output in the **DEBUG CONSOLE** as shown below:


Applications
Merge_Two_Sorted_Link...
Mon 20 Nov, 19:56 prakharguptasim
Merge_Two_Sorted_Linked_List.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MYPROG
Balanced_Array.js
Circular_Linked_List.js
Doubly_Circular_Linked_List.js
Doubly_Linked_List.js
FindMissingNumber.js
Maximum_Subarray_Sum.js
Merge_Two_Sorted_Linked_List.js
Singly_Linked_List.js
Working_with_2D_Array.js
OUTLINE
Merge_Two_Sorted_Linked_List.js > mergeSortedLists
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Filter (e.g. text, !exclude)
Ln 28, Col 2 Spaces: 4 UTF-8 LF {} JavaScript

```
            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;
}
```

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

By following these steps, you have efficiently combined two sorted linked lists into a single sorted list, which helps understand merging techniques used in algorithms for efficient list manipulation. This implementation is concise and leverages the existing order of the lists to minimize operations.