

Lesson 02 Demo 08

Implementing CRUD Operations on a Circular Doubly Linked List

Objective: To implement a circular doubly linked list in JavaScript with CRUD operations including node addition, traversal, value modification, and deletion, enhancing your understanding of advanced linked list structures

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

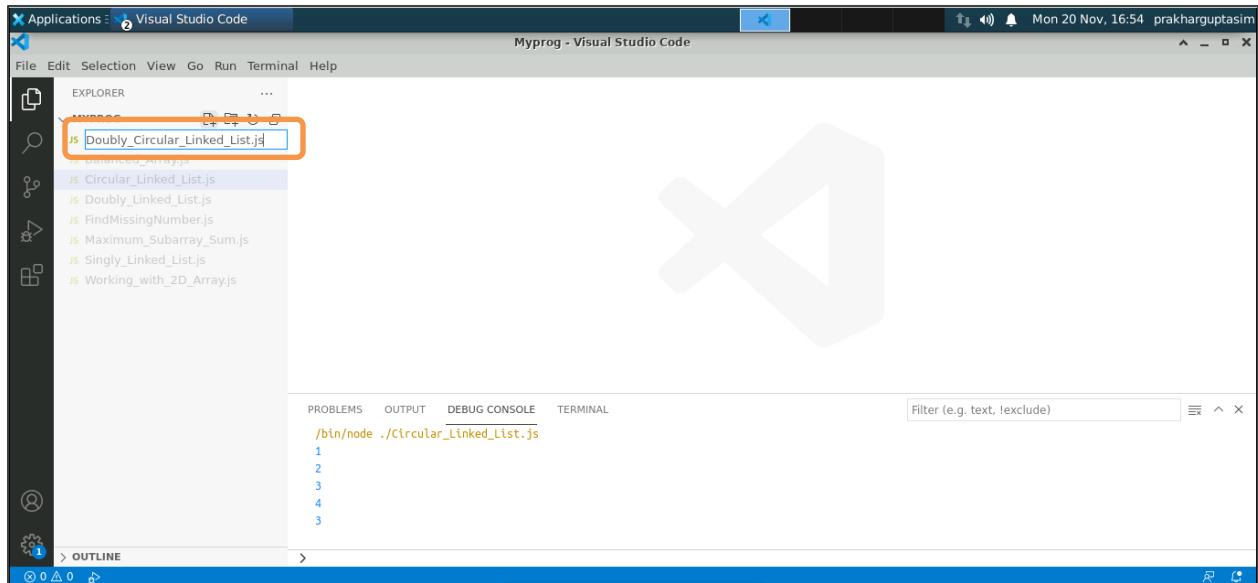
Prerequisites: Completion of Lesson 02 Demo 01

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 **Doubly_Circular_Linked_List.js**



1.2 Add the following code to the file:

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

class DoublyCircularLinkedList {
    constructor() {
        this.head = null;
    }

    // Create: Add a new node to the list
    add(data) {
        const newNode = new ListNode(data);
        if (!this.head) {
            this.head = newNode;
            newNode.next = newNode;
            newNode.prev = newNode;
        } else {
            newNode.prev = this.head.prev;
            newNode.next = this.head;
            this.head.prev.next = newNode;
            this.head.prev = newNode;
        }
    }
}

// Read: Traverse and display elements of the list
read() {
    if (!this.head) {
        return;
    }

    let current = this.head;
    do {
```

```
        console.log(current.data);
        current = current.next;
    } while (current !== this.head);
}

// Update: Modify the value of a node at a given position
update(position, data) {
    if (!this.head) {
        return;
    }

    let current = this.head;
    let count = 0;
    do {
        if (count === position) {
            current.data = data;
            return;
        }
        current = current.next;
        count++;
    } while (current !== this.head);

    console.log("Position not found");
}

// Delete: Remove a node from the list at a specified position
delete(position) {
    if (!this.head) {
        return;
    }

    if (this.head.next === this.head) {
        if (position === 0) {
            this.head = null;
        }
        return;
    }
}
```

```
let current = this.head;
let count = 0;
do {
    if (count === position) {
        current.prev.next = current.next;
        current.next.prev = current.prev;

        if (position === 0) {
            this.head = current.next;
        }
        return;
    }
    current = current.next;
    count++;
} while (current !== this.head);

console.log("Position not found");
}

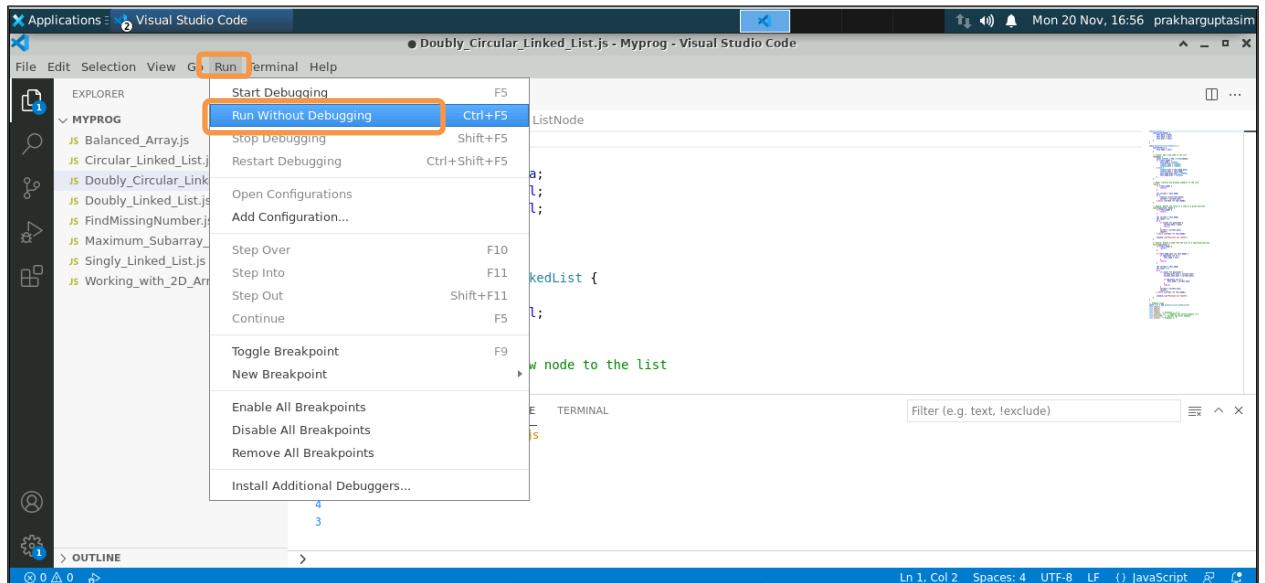
}

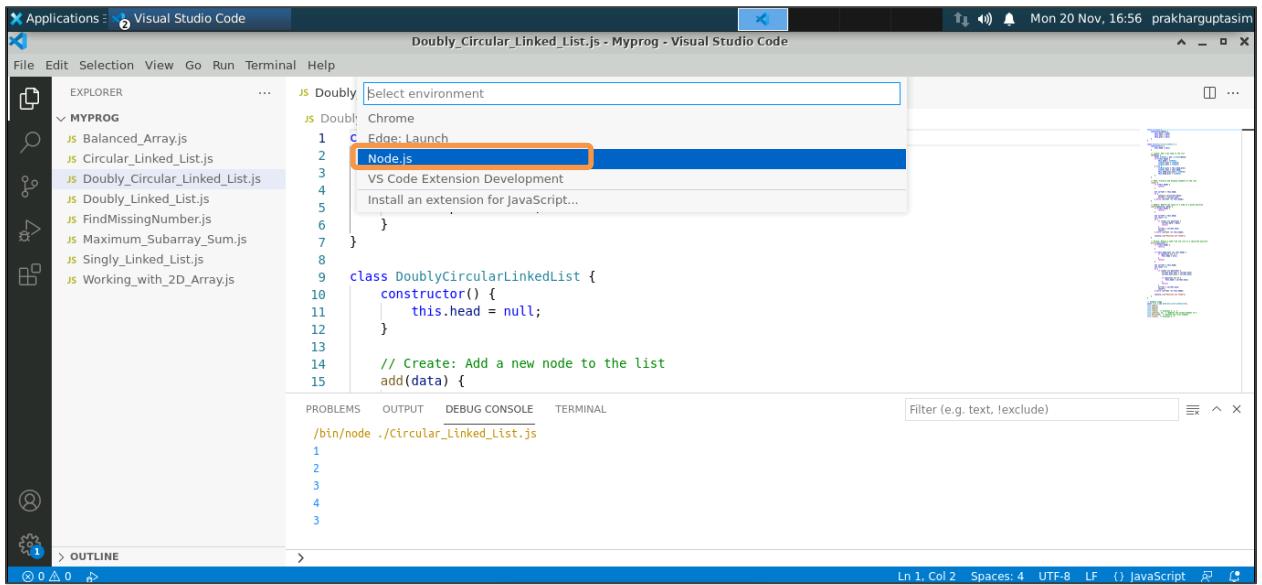
// Example usage
const list = new DoublyCircularLinkedList();
list.add(1);
list.add(2);
list.add(3);
list.read(); // Displays 1, 2, 3
list.update(1, 4); // Updates the second element to 4
list.delete(0); // Deletes the first element
list.read(); // Displays 4, 3
```

The screenshot shows a Visual Studio Code window with the following details:

- Title Bar:** Applications > Visual Studio Code
- File Explorer (Left):** Shows a file tree under the 'MYPROG' folder, including files like Balanced_Array.js, Circular_Linked_List.js, Doubly_Circular_Linked_List.js (selected), Doubly_Linked_List.js, FindMissingNumber.js, Maximum_Subarray_Sum.js, Singly_Linked_List.js, and Working_with_2D_Array.js.
- Code Editor (Center):** Displays the contents of Doubly_Circular_Linked_List.js. The code defines a `ListNode` class and a `DoublyCircularLinkedList` class. It includes a constructor for `ListNode` and a constructor for `DoublyCircularLinkedList` that initializes the head to null. A detailed comment explains the logic for adding a new node to the list, involving setting the new node's next and previous pointers to the current head and the previous node respectively, and adjusting the previous node's next pointer to point to the new node.
- Right Panel:** Shows a preview of the code's output or a related document.
- Bottom Status Bar:** Shows file information (Ln 14, Col 42), encoding (UTF-8), line endings (LF), and the language (JavaScript).

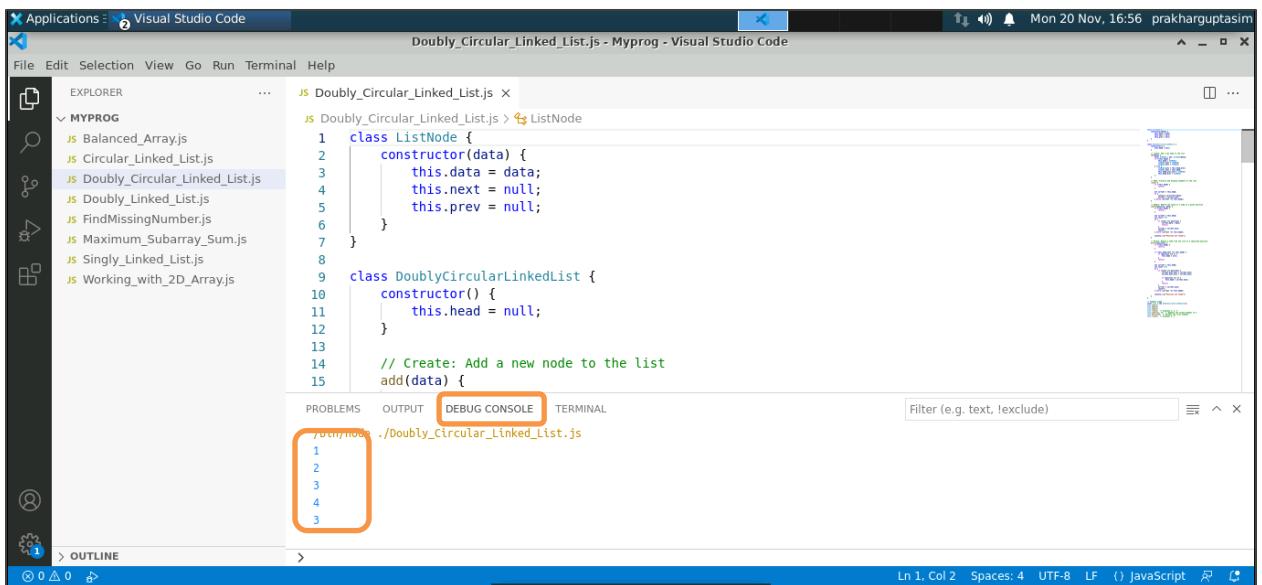
1.3 Click **Run** and then **Run Without Debugging**. Select **Node.js** to check the output in the DEBUG CONSOLE.





The screenshot shows the Visual Studio Code interface with the title bar "Doubly_Circular_Linked_List.js - Myprog - Visual Studio Code". The "File" menu is open, and the "Run" submenu is displayed. The "Node.js" option is highlighted with a red box. The main editor area contains JavaScript code for a doubly circular linked list. The bottom status bar shows "Ln 1, Col 2, Spaces: 4, UTF-8, LF, () JavaScript".

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



The screenshot shows the Visual Studio Code interface with the title bar "Doubly_Circular_Linked_List.js - Myprog - Visual Studio Code". The "File" menu is open, and the "Run" submenu is displayed. The "Node.js" option is highlighted with a red box. The main editor area contains JavaScript code for a doubly circular linked list. The bottom status bar shows "Ln 1, Col 2, Spaces: 4, UTF-8, LF, () JavaScript". The "DEBUG CONSOLE" tab is selected, and it displays the output of the Node.js application, which shows the numbers 1, 2, 3, 4, and 3. A red box highlights the "DEBUG CONSOLE" tab.

By following these steps, you have successfully executed **CRUD** operations on a circular doubly linked list, enhancing your understanding of advanced linked list structures. The **add()** method inserts a new node at the end, the **read()** method traverses and prints the list, the **update()** method modifies the value at a specified position, and the **delete()** method removes a node from the given position.