.

Lesson 02 Demo 06

Implement CRUD Operations on a Doubly Linked List

Objective: To create a doubly linked list in JavaScript with CRUD functionalities such as node addition, traversal, value modification, and node deletion

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

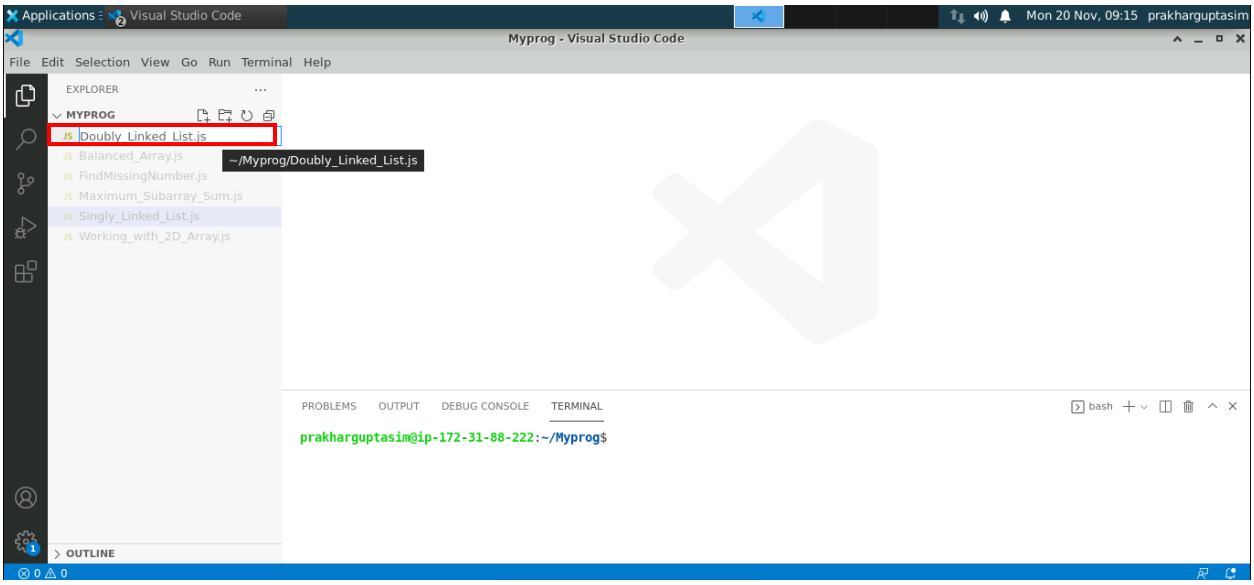
Prerequisites: Perform demo 01 of lesson 02

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 **Doubly_Linked_List.js** as shown below:

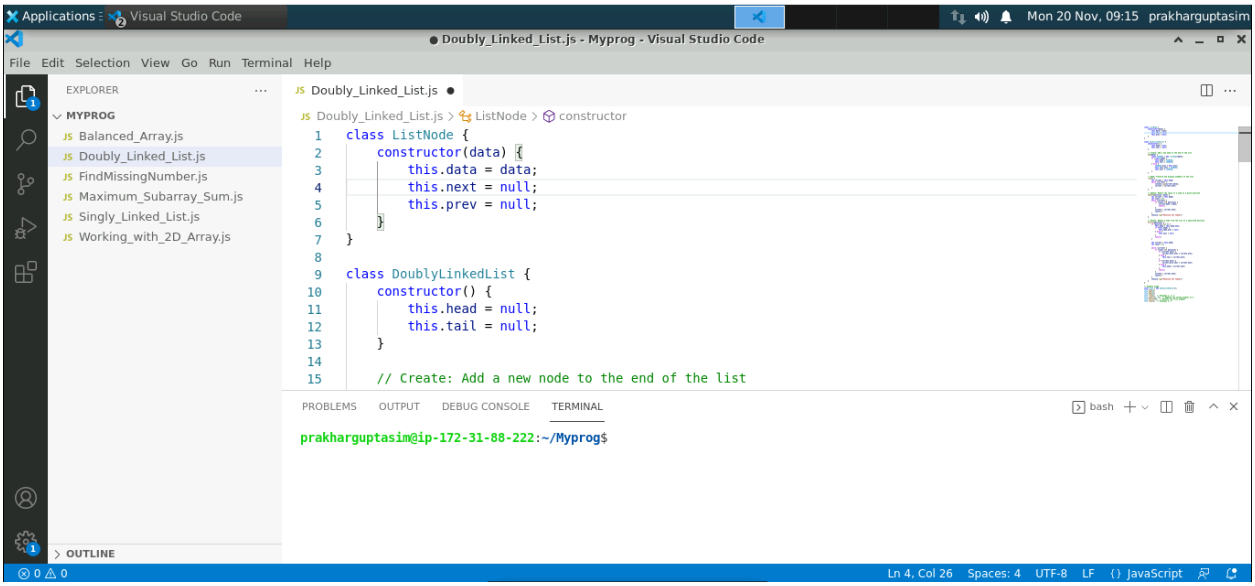


1.2 Paste the code in the file created in step 1.1 as shown below:

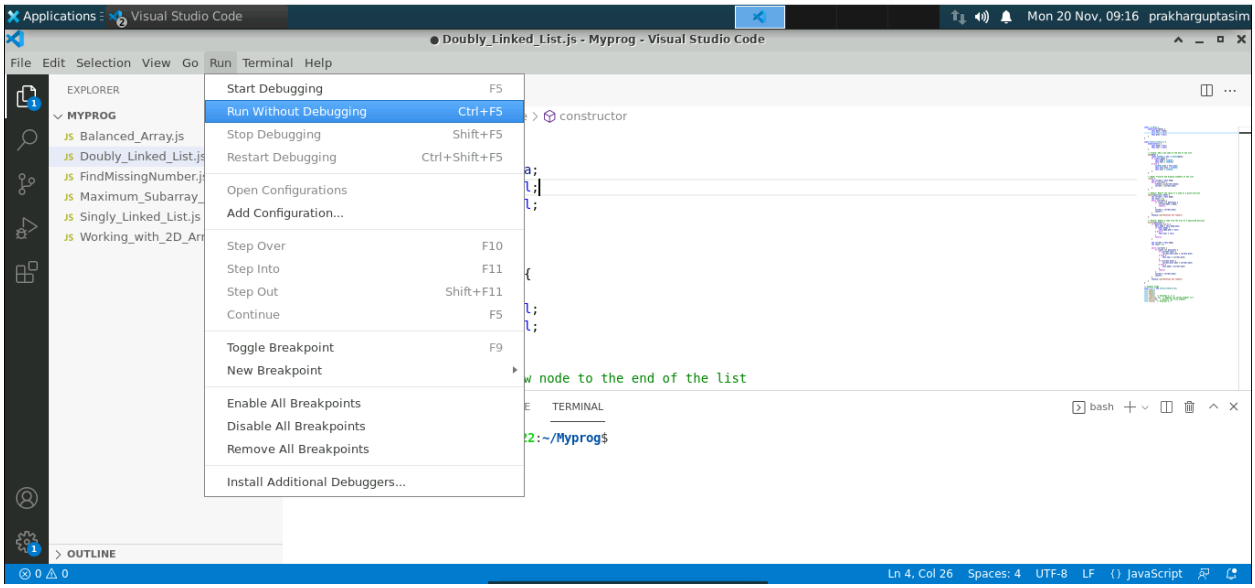
```
class ListNode {
  constructor(data) {
    this.data = data;
    this.next = null;
    this.prev = null;
  }
}

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

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

  // Read: Traverse and display elements of the list
  read() {
    let current = this.head;
    while (current) {
      console.log(current.data);
      current = current.next;
    }
  }
```

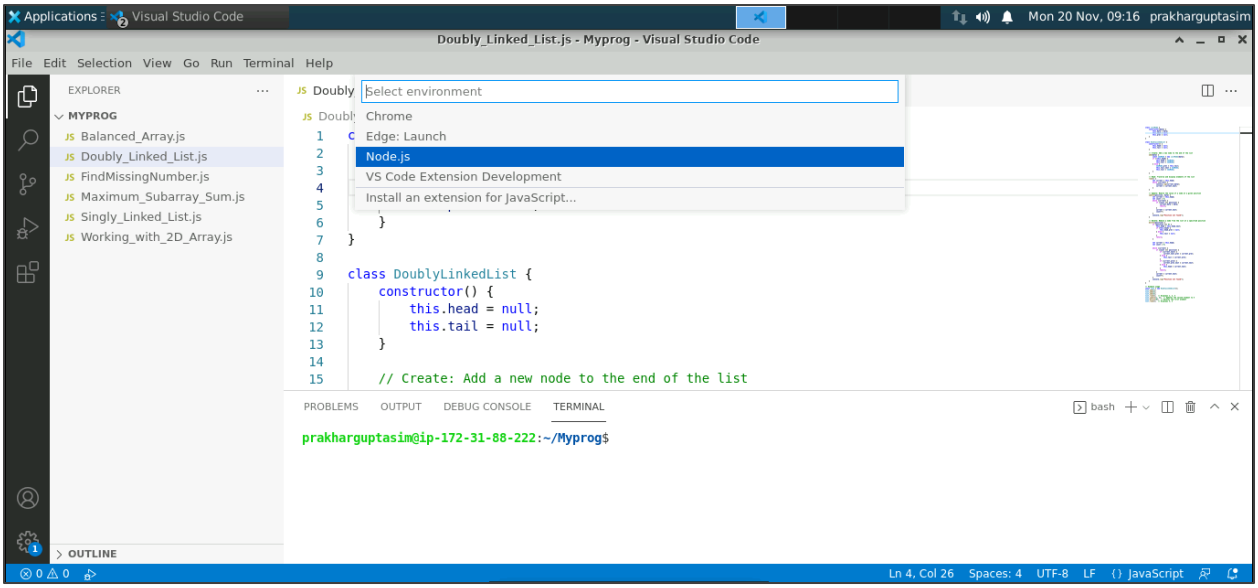
```
// Update: Modify the value of a node at a given position
update(position, data) {
  let current = this.head;
  let count = 0;
  while (current) {
    if (count === position) {
      current.data = data;
      return;
    }
    current = current.next;
    count++;
  }
  console.log("Position not found");
}

// Delete: Remove a node from the list at a specified position
delete(position) {
  if (position === 0) {
    this.head = this.head.next;
    if (this.head) {
      this.head.prev = null;
    } else {
      this.tail = null;
    }
    return;
  }

  let current = this.head;
  let count = 0;

  while (current) {
    if (count === position) {
      if (current.next) {
        current.next.prev = current.prev;
      } else {
        this.tail = current.prev;
      }
```

```
            if (current.prev) {
                current.prev.next = current.next;
            } else {
                this.head = current.next;
            }
            return;
        }
        current = current.next;
        count++;
    }
    console.log("Position not found");
  }
}

// Example usage
const list = new DoublyLinkedList();
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
```

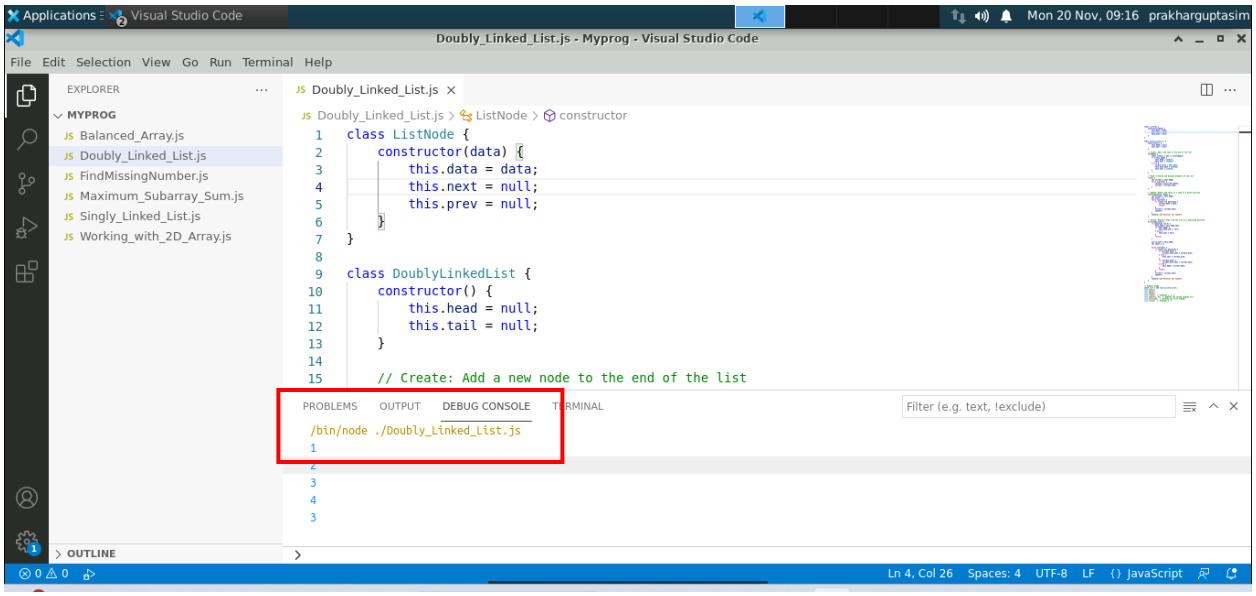


1.3 Save the code and click on **Run->Run Without Debugging->Node.js** to check the output in the debug console





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



By following the above steps, you have successfully performed the **CRUD** operations on a doubly linked list. Here, the **add()** method adds a new node at the end of the list, **read()** method traverses and prints the list, **update()** method changes the value at a given position, and **delete**() method removes a node at a specified position.