

Lesson 02 Demo 05

Implement CRUD Operations on a Singly Linked List

Objective: To create a singly 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

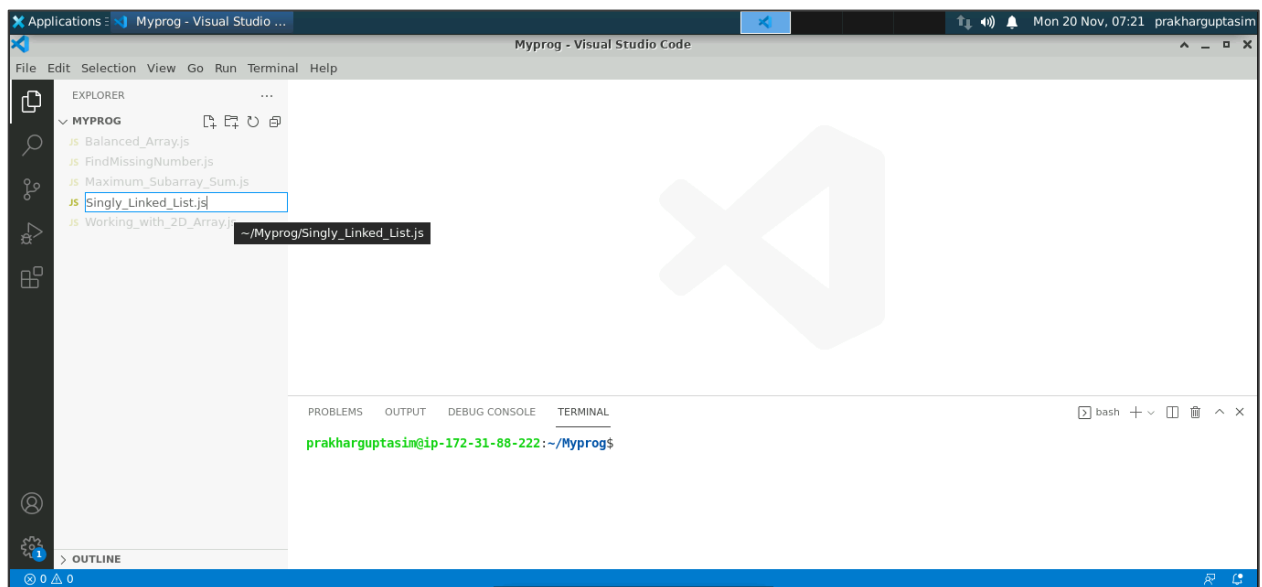
Prerequisites: Perform demo 01 of lesson 02

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 **Singly_Linked_List.js** as shown below:



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

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

class SinglyLinkedList {
  constructor() {
    this.head = 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;
    } else {
      let current = this.head;
      while (current.next) {
        current = current.next;
      }
      current.next = newNode;
    }
  }

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

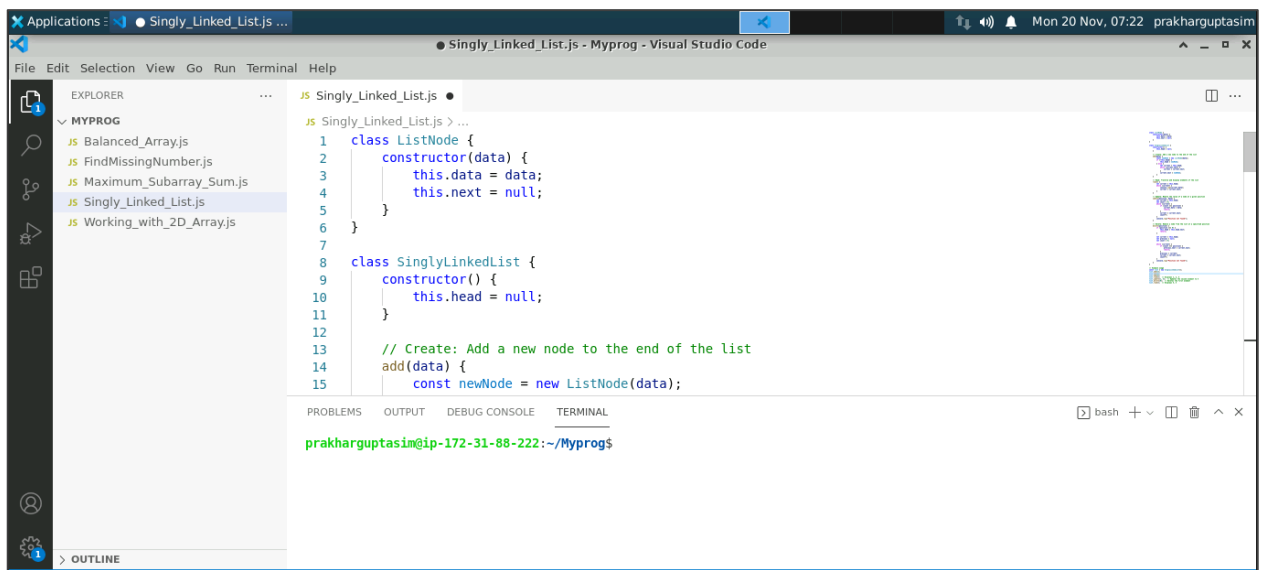
```
// Update: Modify the value of a node at a given position
update(position, data) {
  let current = this.head;
  let count = 0;
  while (current) {
    if (count === position) {
      current.data = data;
      return;
    }
    current = current.next;
    count++;
  }
  console.log("Position not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

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

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

  while (current) {
    if (count === position) {
      previous.next = current.next;
      return;
    }
    previous = current;
    current = current.next;
    count++;
  }
  console.log("Position not found");
}
```

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

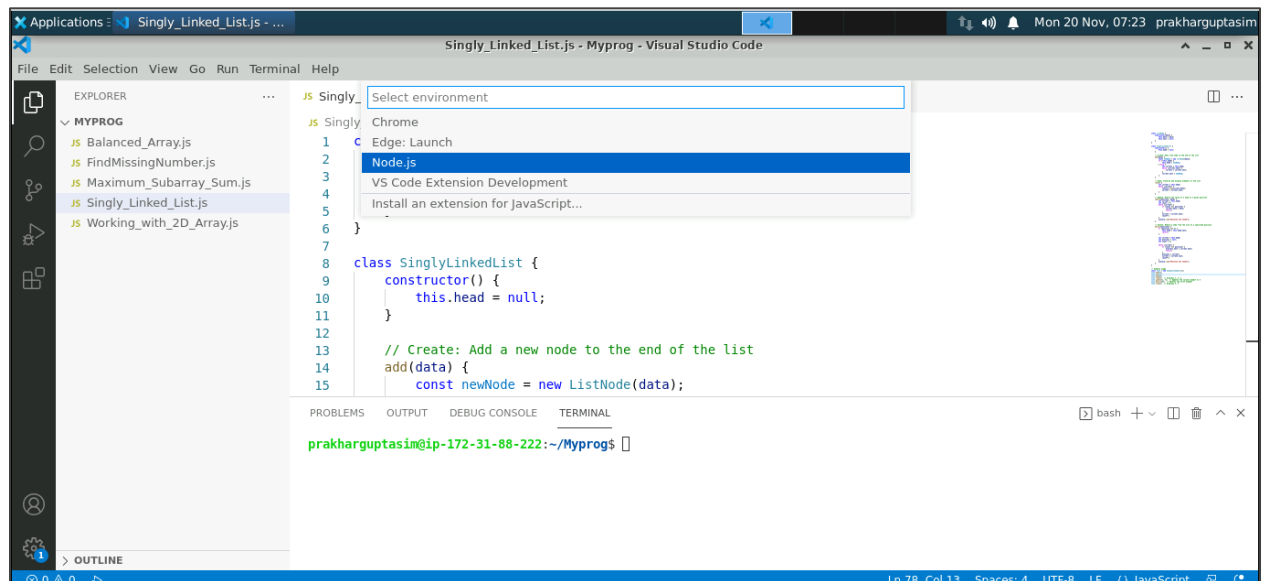
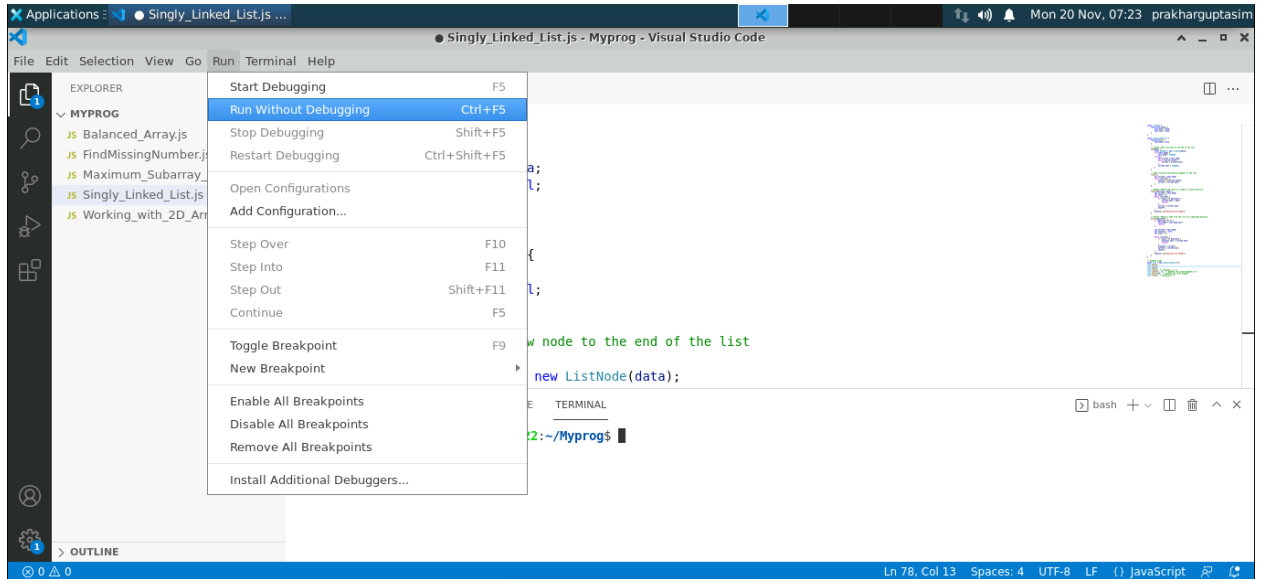


The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'MYPROG' with several JavaScript files, including 'Singly_Linked_List.js'. The code editor displays the implementation of the SinglyLinkedList class. The terminal shows the command prompt for the user 'prakharguptasim' in the directory '~/Myprog'.

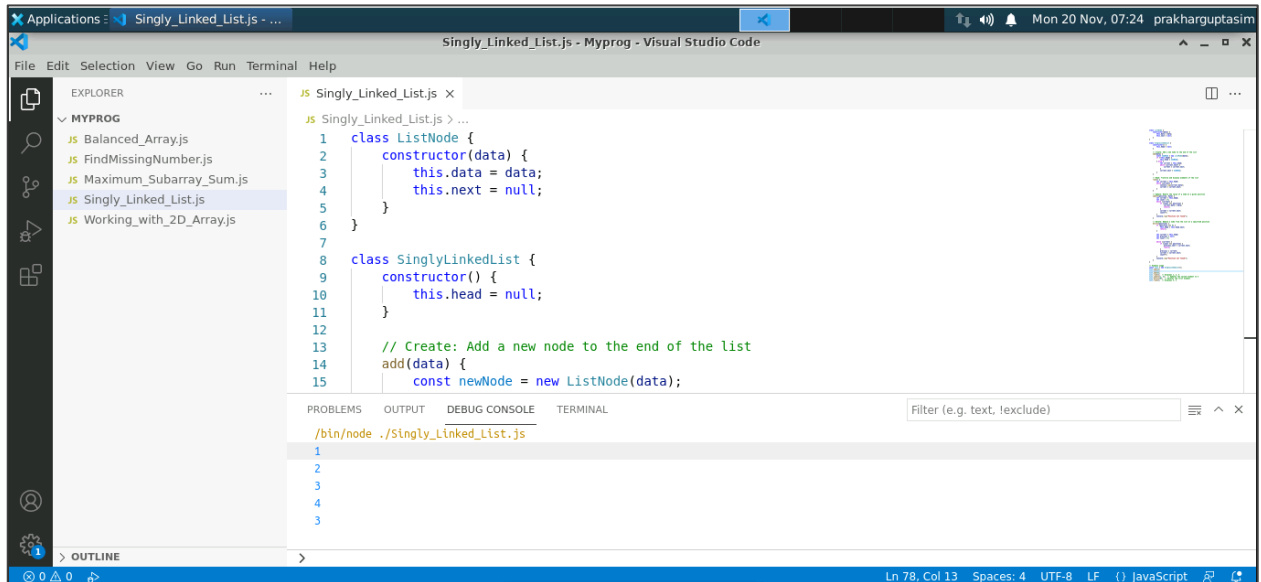
```
JS Singly_Linked_List.js
1 class ListNode {
2   constructor(data) {
3     this.data = data;
4     this.next = null;
5   }
6 }
7
8 class SinglyLinkedList {
9   constructor() {
10    this.head = null;
11  }
12
13  // Create: Add a new node to the end of the list
14  add(data) {
15    const newNode = new ListNode(data);
```

prakharguptasim@ip-172-31-88-222:~/Myprog\$

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



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



The screenshot shows the Visual Studio Code editor with a file named `Singly_Linked_List.js` open. The file contains the following JavaScript code:

```
1 class ListNode {
2   constructor(data) {
3     this.data = data;
4     this.next = null;
5   }
6 }
7
8 class SinglyLinkedList {
9   constructor() {
10    this.head = null;
11  }
12
13  // Create: Add a new node to the end of the list
14  add(data) {
15    const newNode = new ListNode(data);
```

The debug console at the bottom shows the output of the command `./bin/node ./Singly_Linked_List.js`, which is:

```
1
2
3
4
3
```

By following the above steps, you have successfully performed the **CRUD** operations on a singly 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.