Lesson 02 Demo 07

Detecting a Cycle in a Linked List

Objective: To determine whether a linked list contains a cycle by implementing a JavaScript solution that uses pointer-based traversal to enhance system stability, prevent resource leakage, and maintain data integrity in software applications

Tools required: Visual Studio Code and JavaScript

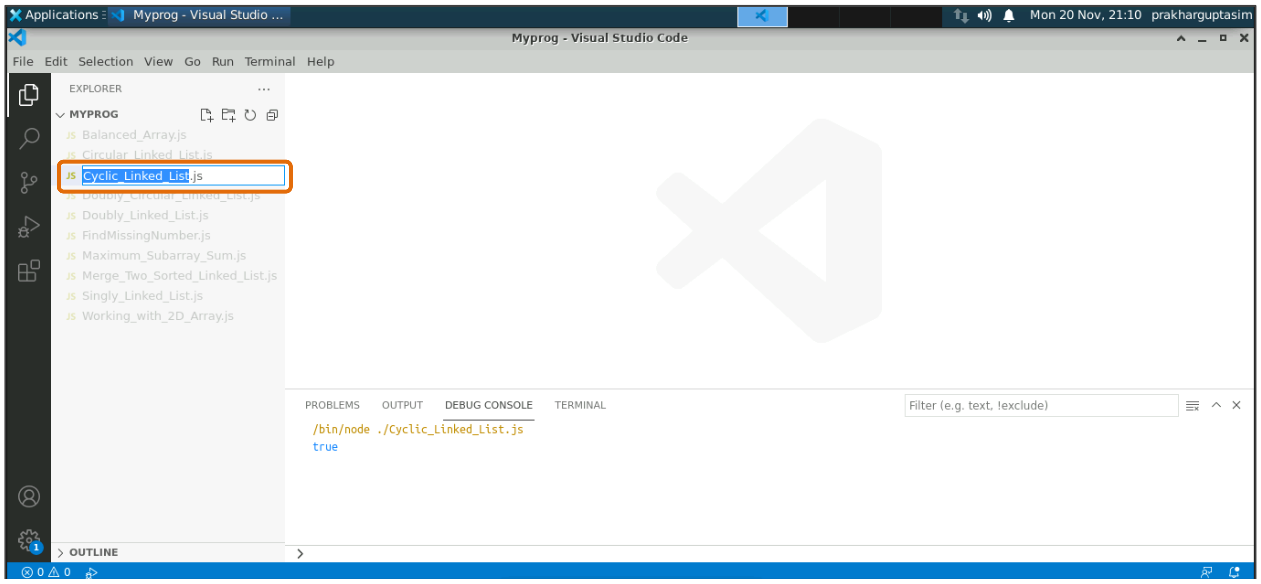
Prerequisites: Completion of Lesson 02 Demo 01

Steps to be followed:

1. Create a JavaScript file and execute it

Step 1: Create a JavaScript file and execute it

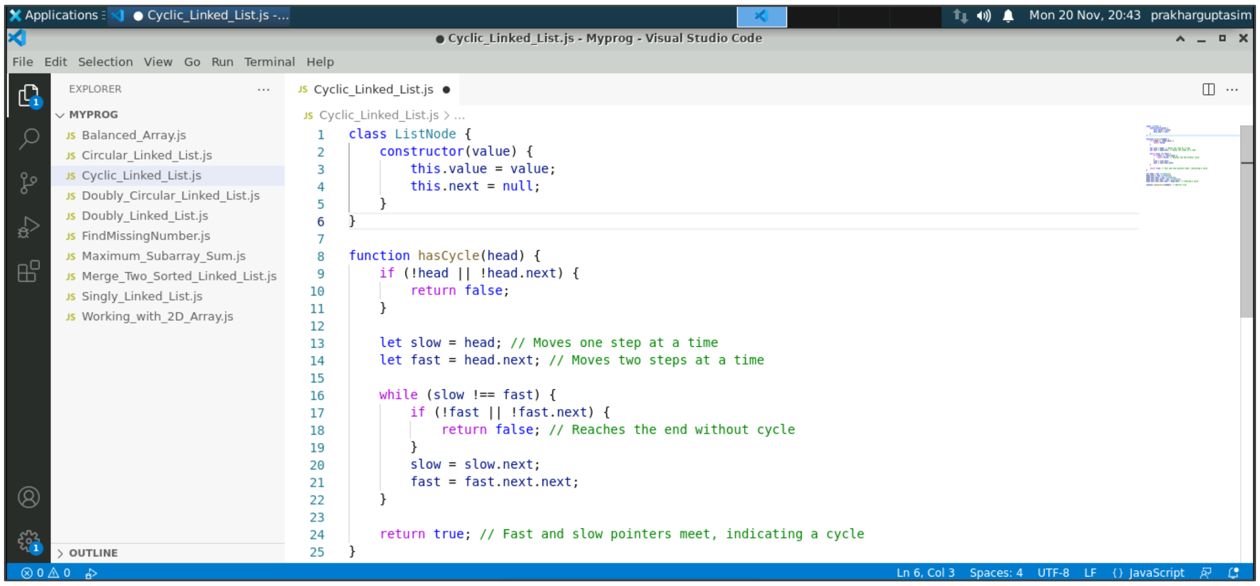
1.1 Open the Visual Studio Code editor and create a JavaScript file named **Cyclic_Linked_List.js**



```
/bin/node ./Cyclic_Linked_List.js
true
```

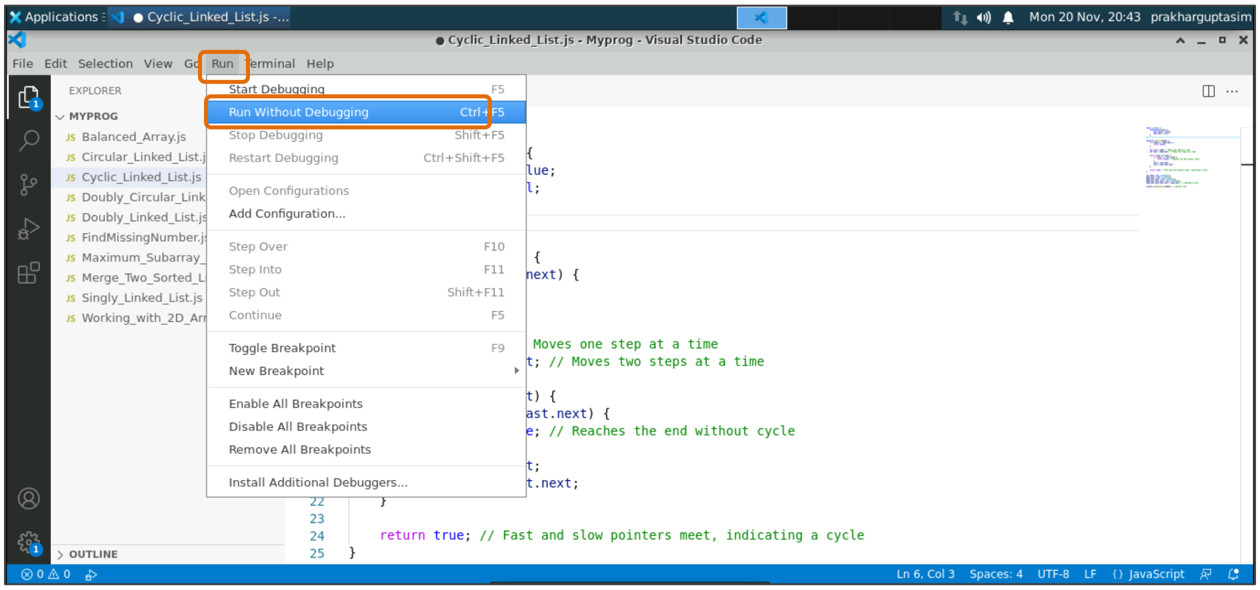
1.2 Add the following code to the file:

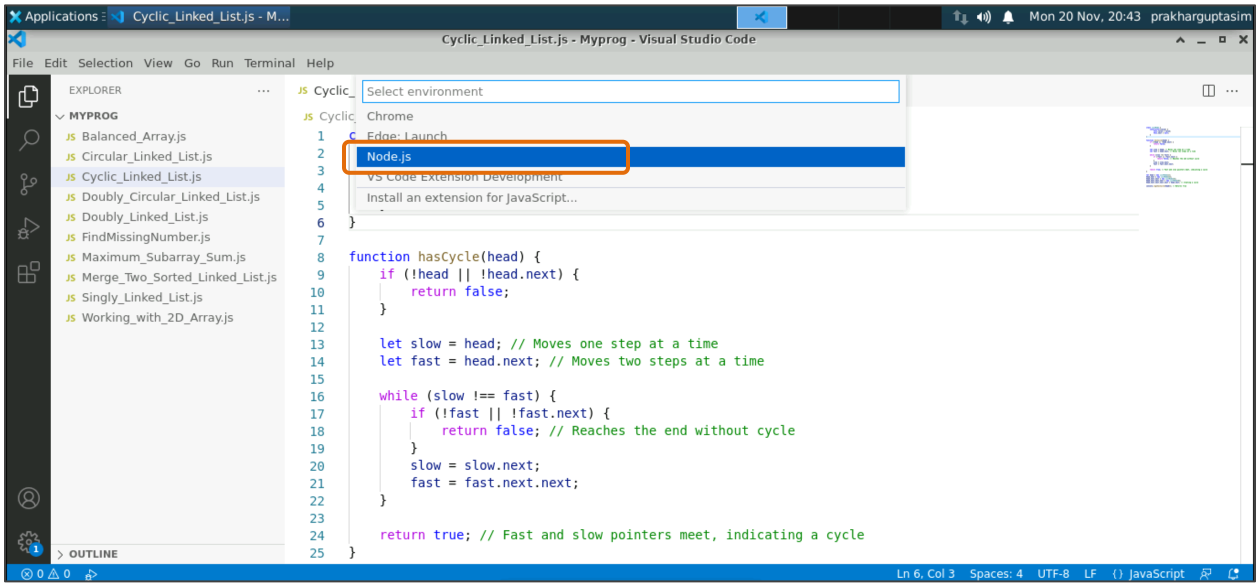
```
   class ListNode {
   constructor(value) {
    this.value = value;
    this.next = null;
    }
   }
  function hasCycle(head) {
  if (!head || !head.next) {
    return false;
   }
  let slow = head; // Moves one step at a time
  let fast = head.next; // Moves two steps at a time
  while (slow !== fast) {
    if (!fast || !fast.next) {
        return false; // Reaches the end without cycle
    }
    slow = slow.next;
    fast = fast.next.next;
 }

  return true; // Fast and slow pointers meet, indicating a cycle
 }
let head = new ListNode(1);
head.next = new ListNode(2);
head.next.next = new ListNode(3);
head.next.next.next = new ListNode(4);
head.next.next.next.next = head.next; // Creating a cycle
console.log(hasCycle(head)); // Returns true
```


Applications
Cyclic_Linked_List.js -...
Mon 20 Nov, 20:43 prakharguptasim
● Cyclic_Linked_List.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MYPROG
Balanced_Array.js
Circular_Linked_List.js
Cyclic_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
Cyclic_Linked_List.js > ...
Ln 6, Col 3 Spaces: 4 UTF-8 LF {} JavaScript

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

function hasCycle(head) {
    if (!head || !head.next) {
        return false;
    }

    let slow = head; // Moves one step at a time
    let fast = head.next; // Moves two steps at a time

    while (slow !== fast) {
        if (!fast || !fast.next) {
            return false; // Reaches the end without cycle
        }
        slow = slow.next;
        fast = fast.next.next;
    }

    return true; // Fast and slow pointers meet, indicating a cycle
}
```

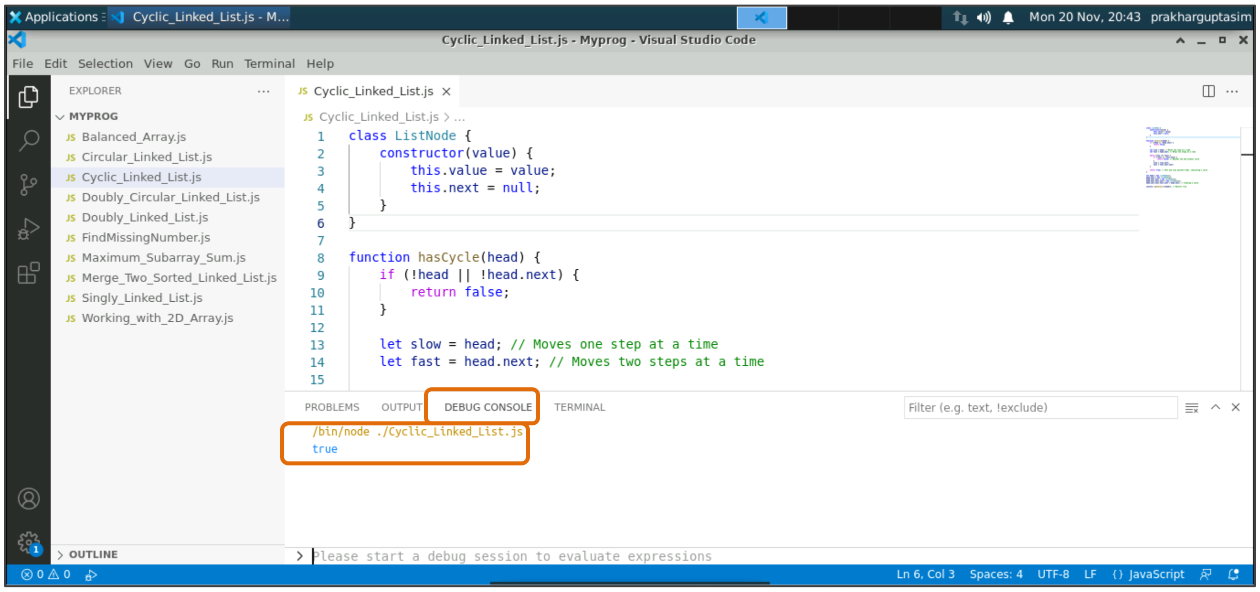
1.3 Click **Run** and then **Run Without Debugging**. Select **Node.js** to check the output in the DEBUG CONSOLE.


Applications
Cyclic_Linked_List.js -...
Mon 20 Nov, 20:43 prakharguptasim
● Cyclic_Linked_List.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
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
Toggle Breakpoint F9
New Breakpoint
Enable All Breakpoints
Disable All Breakpoints
Remove All Breakpoints
Install Additional Debuggers...
Ln 6, Col 3 Spaces: 4 UTF-8 LF {} JavaScript


Cyclic_Linked_List.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MYPROG
Balanced_Array.js
Circular_Linked_List.js
Cyclic_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
Select environment
Chrome
Edge: Launch
Node.js
VS Code Extension Development
Install an extension for JavaScript...
Ln 6, Col 3 Spaces: 4 UTF-8 LF {} JavaScript

```
}

function hasCycle(head) {
    if (!head || !head.next) {
        return false;
    }

    let slow = head; // Moves one step at a time
    let fast = head.next; // Moves two steps at a time

    while (slow !== fast) {
        if (!fast || !fast.next) {
            return false; // Reaches the end without cycle
        }
        slow = slow.next;
        fast = fast.next.next;
    }

    return true; // Fast and slow pointers meet, indicating a cycle
}
```

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


Cyclic_Linked_List.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MYPROG
Balanced_Array.js
Circular_Linked_List.js
Cyclic_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
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Filter (e.g. text, !exclude)
Please start a debug session to evaluate expressions
Ln 6, Col 3 Spaces: 4 UTF-8 LF {} JavaScript

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

function hasCycle(head) {
    if (!head || !head.next) {
        return false;
    }

    let slow = head; // Moves one step at a time
    let fast = head.next; // Moves two steps at a time

```

```
/bin/node ./Cyclic_Linked_List.js
true
```

By following these steps, you have successfully implemented a JavaScript solution to detect cycles in a linked list. This helps enhance system stability, prevent resource leakage, and maintain data integrity in software applications.