Name of Student: Naveen Gummella			
Roll Number: 16		**LAB Assignment Number:** 2	
Title of LAB Assignment: Create an application to demonstrate Node.js Modules			
DOP: 09/09/23		**DOS:** 15/09/2023	
CO Mapped: CO1	**PO Mapped:** PO3,PO5,PSO1, PSO2	**Signature:** Naveen. G	**Marks:**

Aim:

Create an application to demonstrate Node.js Module:

1. Built in module-

   i) Write a program to print information about the computer's operating system using the OS module (use any 5 methods).

   ii) Print "Hello" every 500 milliseconds using the Timer Module. The message should be printed exactly 10 times. Use SetInterval, ClearInterval and SetTimeout methods.
2. Custom module-

   i) create a Calculator Node.js Module with functions add, subtract and multiply, Divide. And use the Calculator module in another Node.js file.

   ii) Create a circle module with functions to find the area and perimeter of a circle and use it.

Description:

Node.js modules are a fundamental part of building modular and maintainable applications in Node.js. They allow you to organize your code into reusable and encapsulated units.

1. What are Node.js Modules?

Node.js modules are individual units of code that encapsulate specific functionality. They help in organizing and structuring your codebase, making it more maintainable and modular. Modules in Node.js can be:

a. Built-in Modules: These are modules that come with Node.js, such as fs for file system operations and http for creating web servers.

b. Third-party Modules: These are modules created by the Node.js community and can be easily installed and managed using npm (Node Package Manager).

c. Custom Modules: These are modules you create to encapsulate your own code and logic. Custom modules can be reused across different parts of your application.

2. Common Module Patterns:

There are several common module patterns in Node.js:

a. CommonJS Modules: This is the default module system in Node.js. You use require to import modules and module.exports to export functionality.

b. ES6 Modules: With the introduction of ES6, Node.js also supports ES6 modules using import and export statements.

3. Benefits of Node.js Modules:

a. Encapsulation: Modules allow you to encapsulate code and data, preventing global scope pollution and naming conflicts.

b. Reusability: You can easily reuse modules across different parts of your application or in other projects.

c. Maintainability: Modules promote code organization, making it easier to maintain and debug your codebase.

d. Dependency Management: With npm, you can manage dependencies efficiently, ensuring that your project uses the correct versions of third-party modules.

4. Module Resolution:

Node.js uses a specific algorithm to resolve modules. It searches for modules in the following locations:

a. Core modules:Modules installed in the node_modules directory of the current module
Parent and ancestor directories' node_modules directories.
This allows you to control module versioning and avoid conflicts.

5. Use Cases:

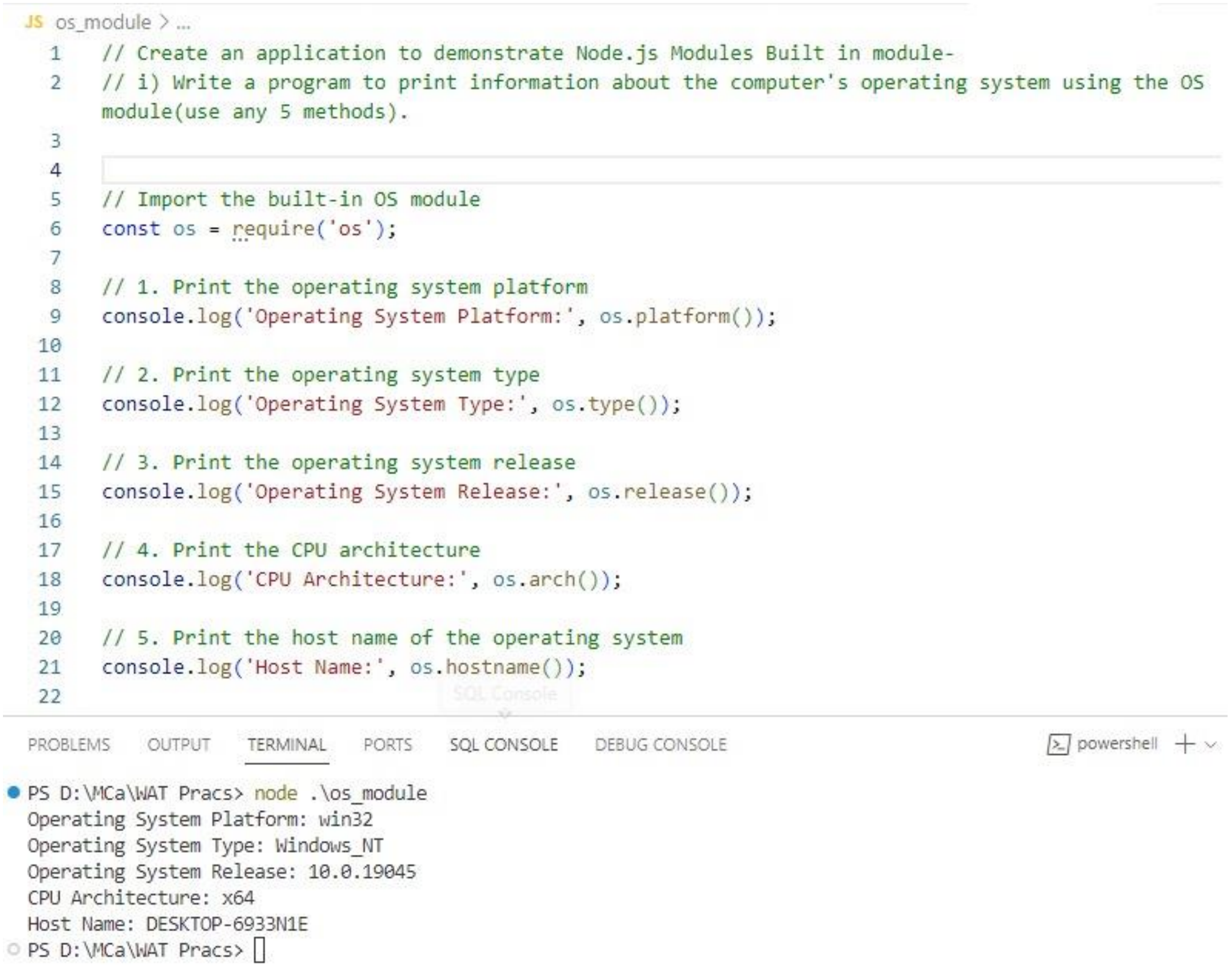
Node.js modules are used in various scenarios, such as:

a. Creating web servers and APIs with modules like http and express.
b. Reading and writing files with modules like fs.
c. Managing asynchronous operations with modules like async/await and promisify.

Code & Output:

Built in module-

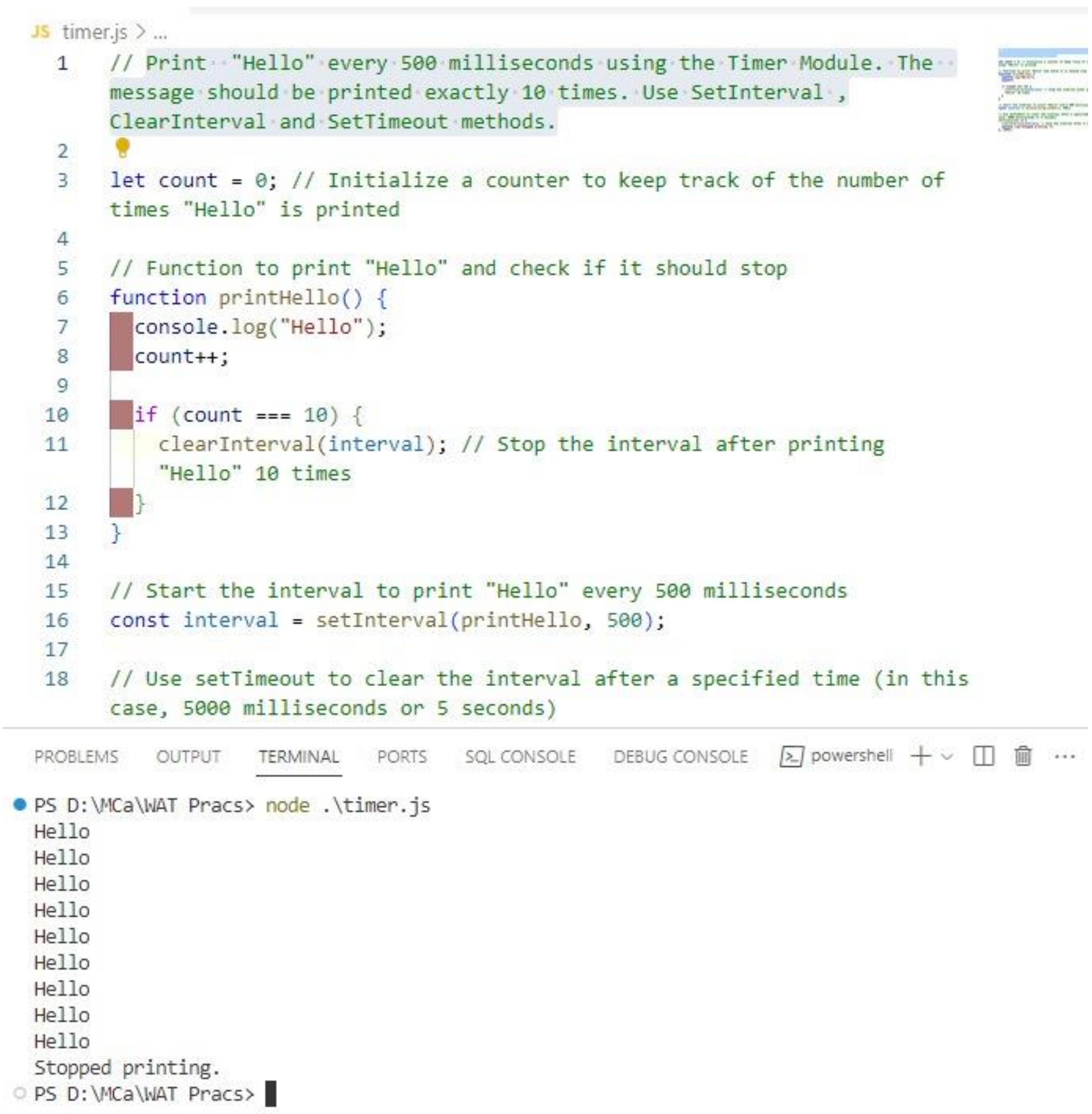
i) Write a program to print information about the computer's operating system using the OS module.



```
JS os_module > ...
// Create an application to demonstrate Node.js Modules Built in module-
// i) Write a program to print information about the computer's operating system using the OS
module(use any 5 methods).


// Import the built-in OS module
const os = require('os');

// 1. Print the operating system platform
console.log('Operating System Platform:', os.platform());

// 2. Print the operating system type
console.log('Operating System Type:', os.type());

// 3. Print the operating system release
console.log('Operating System Release:', os.release());

// 4. Print the CPU architecture
console.log('CPU Architecture:', os.arch());

// 5. Print the host name of the operating system
console.log('Host Name:', os.hostname());

```

```
PS D:\MCa\WAT Pracs> node .\os_module
Operating System Platform: win32
Operating System Type: Windows_NT
Operating System Release: 10.0.19045
CPU Architecture: x64
Host Name: DESKTOP-6933N1E
PS D:\MCa\WAT Pracs>
```

ii) Print "Hello" every 500 milliseconds using the Timer Module. The message should be printed exactly 10 times. Use SetInterval ,ClearInterval and SetTimeout methods.

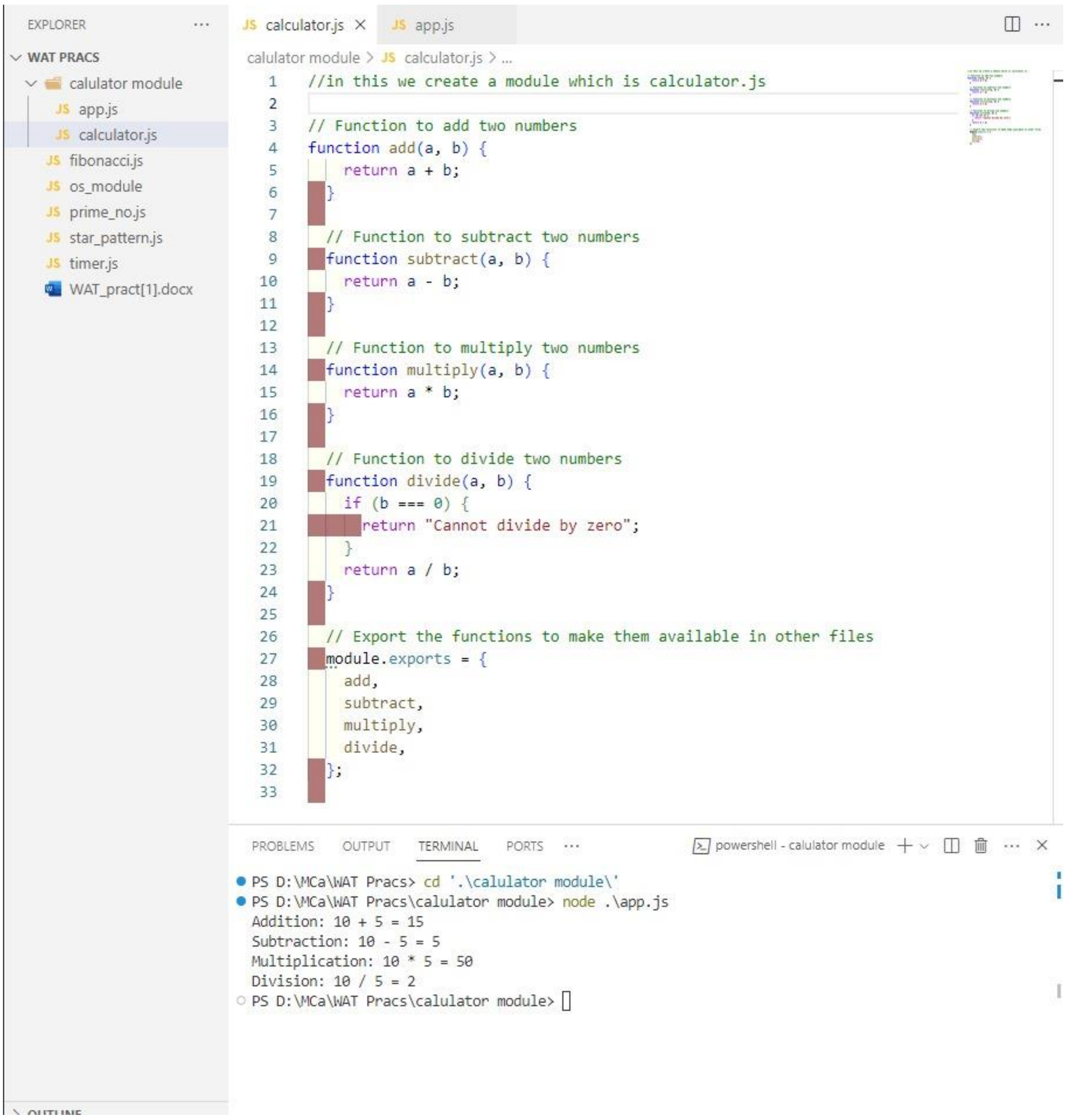


```javascript
// Print  "Hello" every 500 milliseconds using the Timer Module. The
message should be printed exactly 10 times. Use SetInterval ,
ClearInterval and SetTimeout methods.

let count = 0; // Initialize a counter to keep track of the number of
times "Hello" is printed

// Function to print "Hello" and check if it should stop
function printHello() {
  console.log("Hello");
  count++;

  if (count === 10) {
    clearInterval(interval); // Stop the interval after printing
    "Hello" 10 times
  }
}

// Start the interval to print "Hello" every 500 milliseconds
const interval = setInterval(printHello, 500);

// Use setTimeout to clear the interval after a specified time (in this
case, 5000 milliseconds or 5 seconds)
```

```
PS D:\MCa\WAT Pracs> node .\timer.js
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Stopped printing.
PS D:\MCa\WAT Pracs>
```

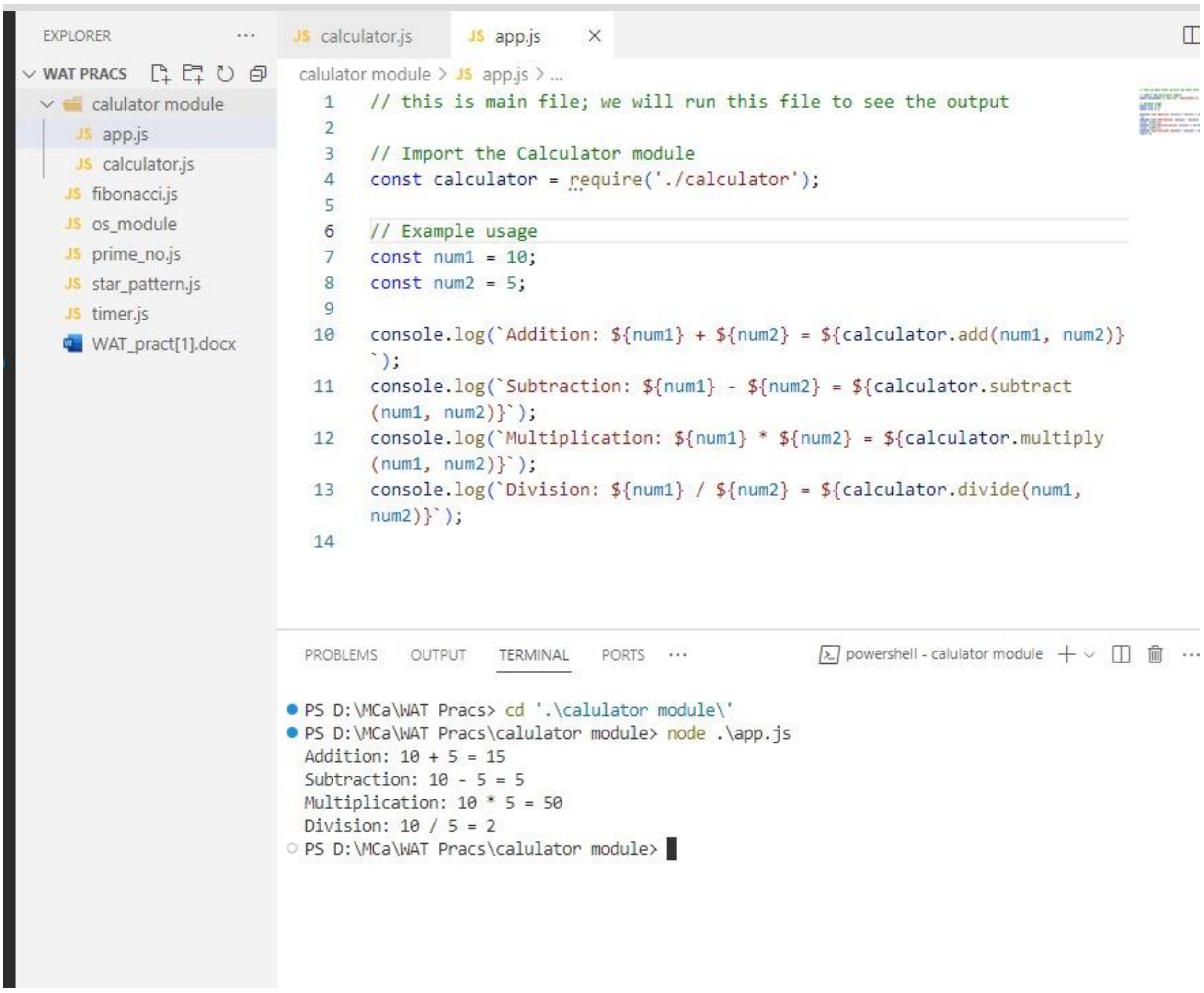
Custom module-

i) create a Calculator Node.js Module with functions add, subtract and multiply,Divide. And use the Calculator module in another Node.js file.



```javascript
//in this we create a module which is calculator.js

// Function to add two numbers
function add(a, b) {
    return a + b;
}

// Function to subtract two numbers
function subtract(a, b) {
    return a - b;
}

// Function to multiply two numbers
function multiply(a, b) {
    return a * b;
}

// Function to divide two numbers
function divide(a, b) {
    if (b === 0) {
        return "Cannot divide by zero";
    }
    return a / b;
}

// Export the functions to make them available in other files
module.exports = {
    add,
    subtract,
    multiply,
    divide,
};
```

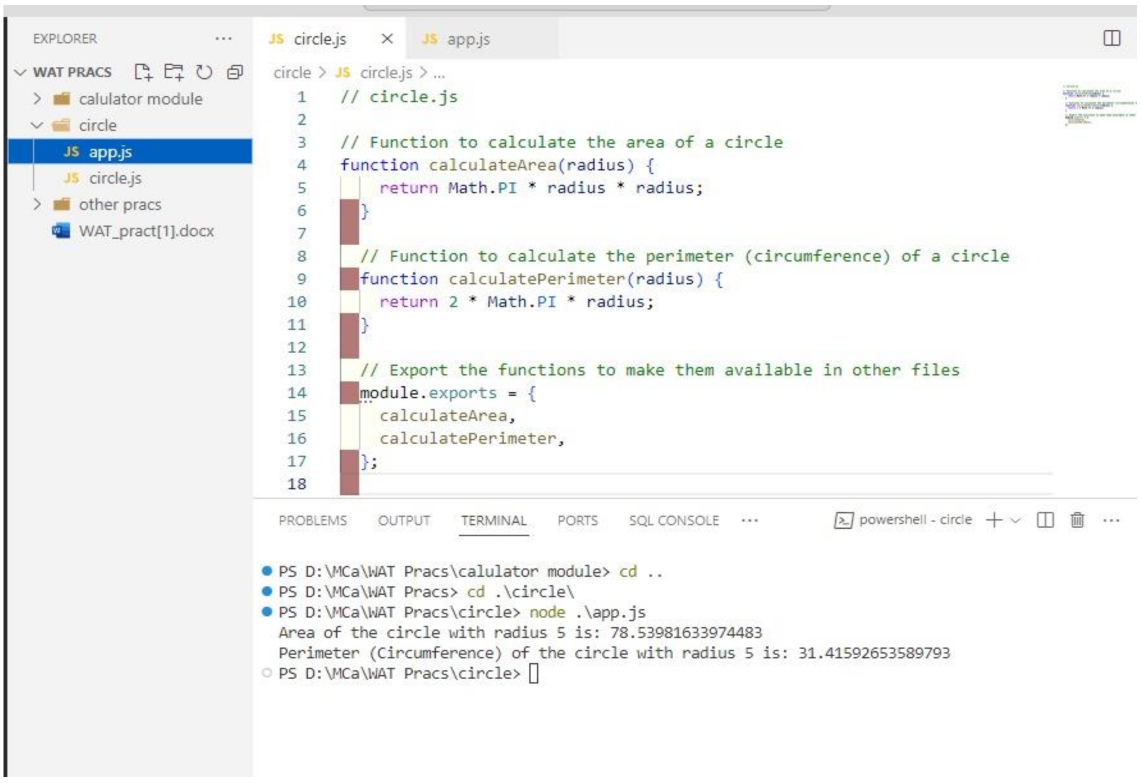
```
PS D:\MCa\WAT Pracs> cd '.\calulator module\'
PS D:\MCa\WAT Pracs\calulator module> node .\app.js
Addition: 10 + 5 = 15
Subtraction: 10 - 5 = 5
Multiplication: 10 * 5 = 50
Division: 10 / 5 = 2
PS D:\MCa\WAT Pracs\calulator module>
```



```javascript
// this is main file; we will run this file to see the output

// Import the Calculator module
const calculator = require('./calculator');

// Example usage
const num1 = 10;
const num2 = 5;

console.log(`Addition: ${num1} + ${num2} = ${calculator.add(num1, num2)}`);
console.log(`Subtraction: ${num1} - ${num2} = ${calculator.subtract(num1, num2)}`);
console.log(`Multiplication: ${num1} * ${num2} = ${calculator.multiply(num1, num2)}`);
console.log(`Division: ${num1} / ${num2} = ${calculator.divide(num1, num2)}`);

```

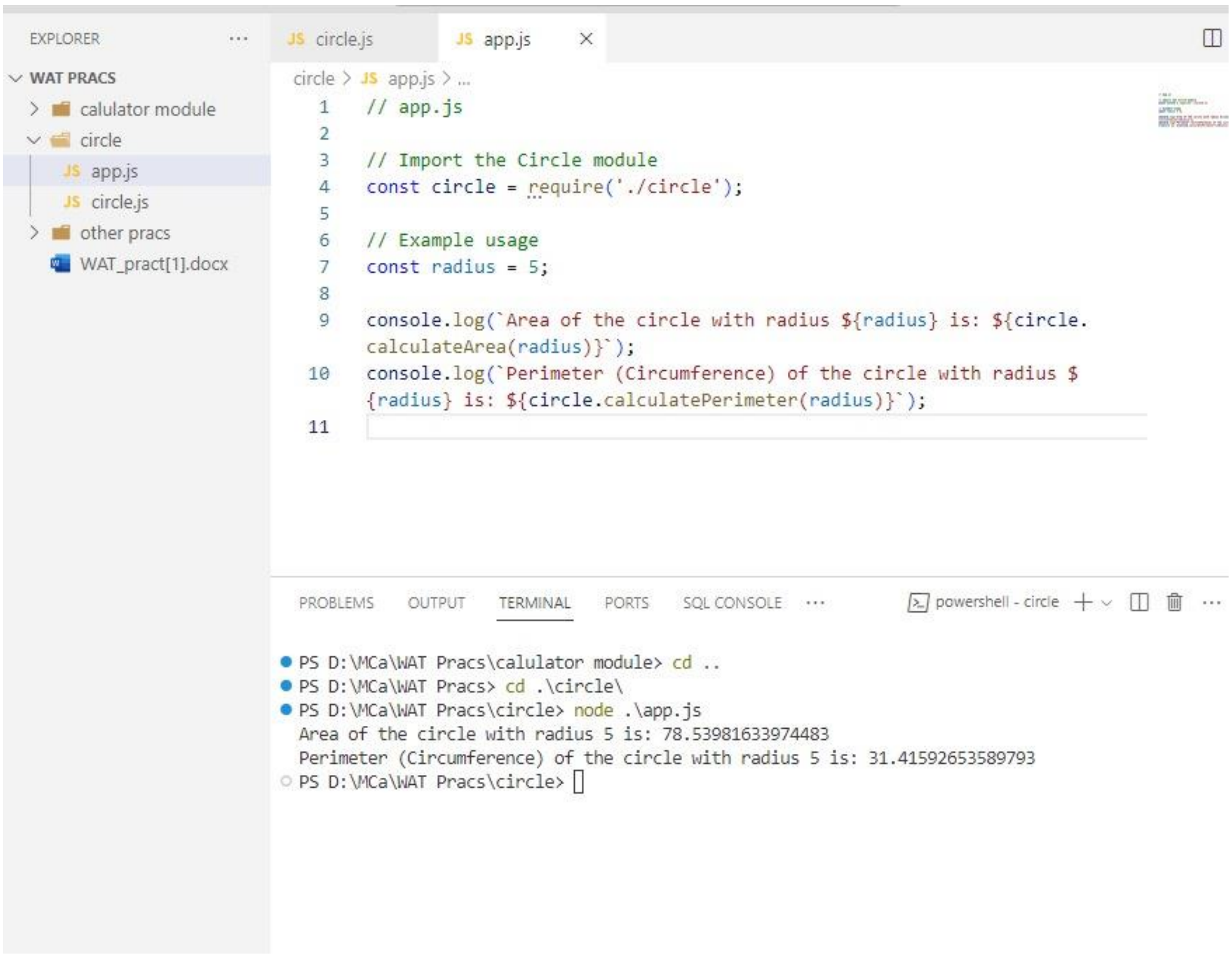
```
PS D:\MCa\WAT Pracs> cd '.\calulator module\'
PS D:\MCa\WAT Pracs\calulator module> node .\app.js
Addition: 10 + 5 = 15
Subtraction: 10 - 5 = 5
Multiplication: 10 * 5 = 50
Division: 10 / 5 = 2
PS D:\MCa\WAT Pracs\calulator module>
```

ii) Create a circle module with functions to find the area and perimeter of a circle and use it.



```javascript
// circle.js

// Function to calculate the area of a circle
function calculateArea(radius) {
    return Math.PI * radius * radius;
}

// Function to calculate the perimeter (circumference) of a circle
function calculatePerimeter(radius) {
    return 2 * Math.PI * radius;
}

// Export the functions to make them available in other files
module.exports = {
    calculateArea,
    calculatePerimeter,
};

```

```
PS D:\MCa\WAT Pracs\calulator module> cd ..
PS D:\MCa\WAT Pracs> cd .\circle\
PS D:\MCa\WAT Pracs\circle> node .\app.js
Area of the circle with radius 5 is: 78.53981633974483
Perimeter (Circumference) of the circle with radius 5 is: 31.41592653589793
PS D:\MCa\WAT Pracs\circle>
```



```
// app.js

// Import the Circle module
const circle = require('./circle');

// Example usage
const radius = 5;

console.log(`Area of the circle with radius ${radius} is: ${circle.
calculateArea(radius)}`);
console.log(`Perimeter (Circumference) of the circle with radius $
{radius} is: ${circle.calculatePerimeter(radius)}`);
```

```
PS D:\MCa\WAT Pracs\calulator module> cd ..
PS D:\MCa\WAT Pracs> cd .\circle\
PS D:\MCa\WAT Pracs\circle> node .\app.js
Area of the circle with radius 5 is: 78.53981633974483
Perimeter (Circumference) of the circle with radius 5 is: 31.41592653589793
PS D:\MCa\WAT Pracs\circle>
```

Conclusion:

Node.js modules are a critical part of building scalable and maintainable applications in Node.js. They provide a way to encapsulate and organize your code, promote reusability, and enable efficient dependency management. Understanding module patterns and how module resolution works is essential for building Node.js applications effectively.