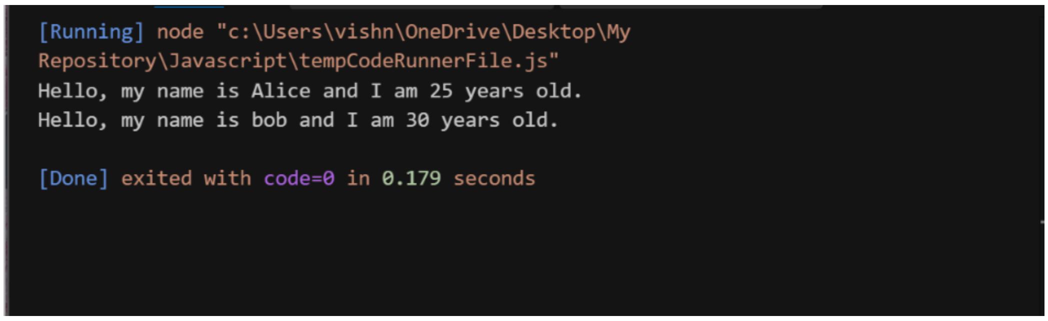
CLASS IN JAVASCRIPT

Problem Statement 1:
Create a Constructor for a Person Write a JavaScript function constructor named Person that takes two parameters, name, and age, and Add a method to the prototype to display name and age. Then, create two instances of Person and display their names and ages

Code:

```
function person(name, age){
    this.name = name;
    this.age = age;

    this.student = function(){
        console.log(`Hello, my name is ${this.name} and I am
${this.age} years old.`);
    };
}
let person1 = new person('Alice', 25);
let person2 = new person('bob', 30);

person1.student();
person2.student();
```

Output:



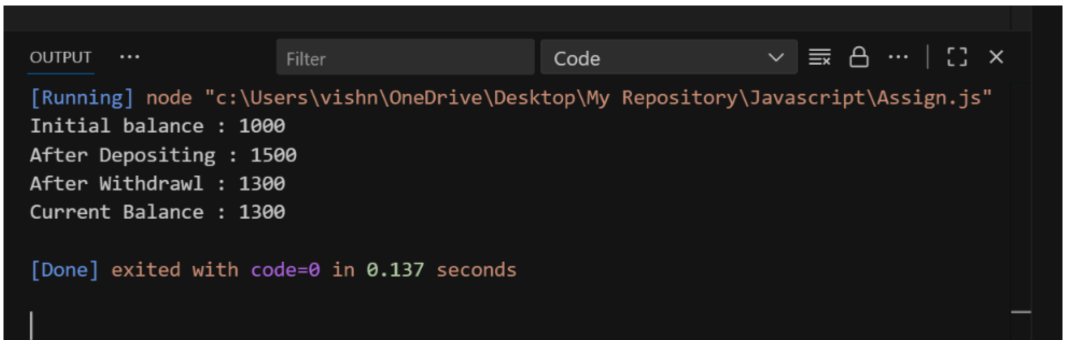
```
[Running] node "c:\Users\vishn\OneDrive\Desktop\My
Repository\Javascript\tempCodeRunnerFile.js"
Hello, my name is Alice and I am 25 years old.
Hello, my name is bob and I am 30 years old.

[Done] exited with code=0 in 0.179 seconds
```

Problem Statement 2: Implement a Bank Account Create a constructor function named BankAccount that initializes a bank account with an initial balance. Include methods for depositing and withdrawing money from the account. Write code to demonstrate these operations on a bank account object.

Code:

```
function backAccount(initialBalance){
    this.balance = initialBalance;

    this.deposit = function(amount){
        this.balance += amount;
        return this.balance;
    };

    this.withdraw = function(amount){
        this.balance -= amount;
        return this.balance;
    };

    this.getBalance = function(){
        return this.balance;
    };
}
let myAccount = new backAccount(1000);
console.log('Initial balance :',myAccount.getBalance());
console.log('After Depositing :',myAccount.deposit(500));
console.log('After Withdraw :',myAccount.withdraw(200));
console.log('Current Balance :',myAccount.getBalance());
```

Output:



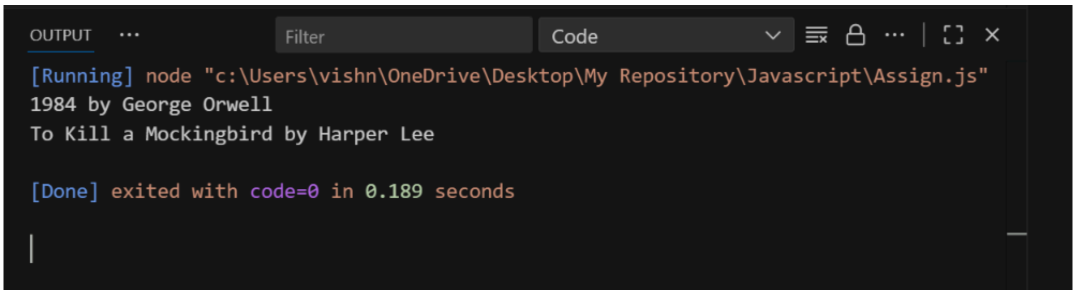
```
[Running] node "c:\Users\vishn\OneDrive\Desktop\My Repository\Javascript\Assign.js"
Initial balance : 1000
After Depositing : 1500
After Withdrawl : 1300
Current Balance : 1300

[Done] exited with code=0 in 0.137 seconds
```

Problem Statement 3: Create a Constructor for a Book Design a constructor function called Book that takes title and author as parameters. Add a method to the

prototype of the Book that displays the book's information. Create at least two book instances and display their information.
Code:

```
function book(title, author){
    this.title = title;
    this.author = author;

    this.bookInfo = function(){
        return this.title + " by " + this.author;
    };
}
let book1 = new book("1984", "George Orwell");
let book2 = new book("To Kill a Mockingbird", "Harper Lee");
console.log(book1.bookInfo());
console.log(book2.bookInfo());
```

Output:



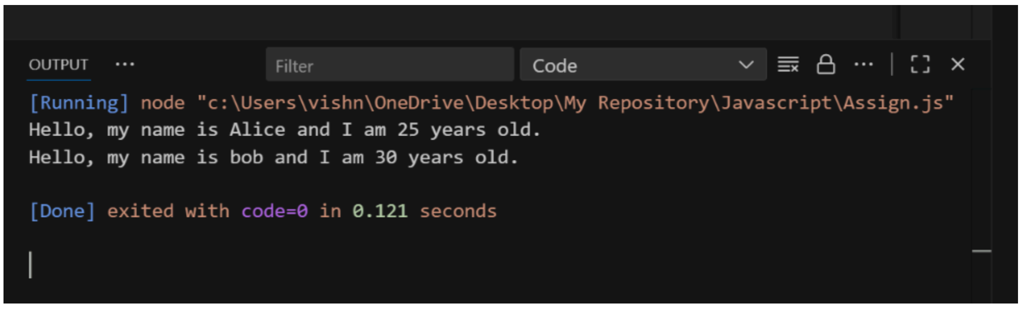
```
[Running] node "c:\Users\vishn\OneDrive\Desktop\My Repository\Javascript\Assign.js"
1984 by George Orwell
To Kill a Mockingbird by Harper Lee

[Done] exited with code=0 in 0.189 seconds
```

Problem Statement 4: Implement Task 1 using Class. Design a JavaScript class called Person with properties for name and age. Implement a method to display the person's name and age. Then, create instances of Person and display their information.

Code:

```
class person {
    constructor(name, age) {
        this.name = name;
        this.age = age;


        this.student = function () {
            console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);
        };
    }
}
let person1 = new person('Alice', 25);
let person2 = new person('bob', 30);


person1.student();
person2.student();
```

Output:



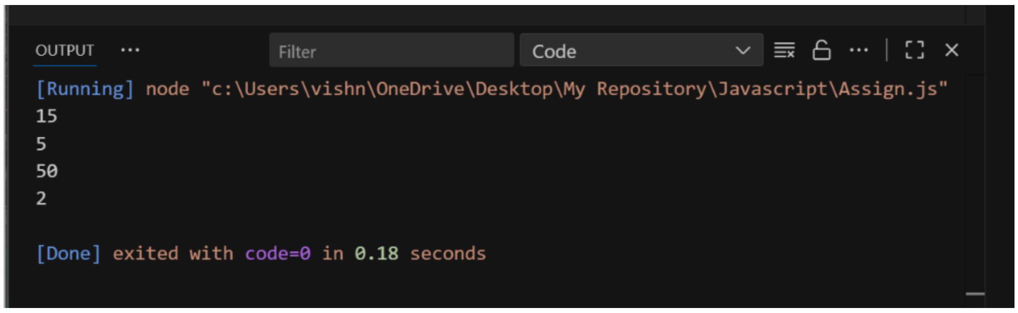
```
[Running] node "c:\Users\vishn\OneDrive\Desktop\My Repository\Javascript\Assign.js"
Hello, my name is Alice and I am 25 years old.
Hello, my name is bob and I am 30 years old.

[Done] exited with code=0 in 0.121 seconds
```

Problem Statement 5: Implement a Calculator Class Create a class called Calculator that initializes two values value1 and value2 to store numbers. Add methods for add, subtract, multiply, and divide. Perform and show operations.

Code:

```
class Calculator {
    num1;
    num2;
    constructor(value1, value2){
        this.num1 = value1;
        this.num2 = value2;
    };
    add(){
        return this.num1 + this.num2;
    }
    subtract(){
        return this.num1 - this.num2;
    }
    multiply(){
        return this.num1 * this.num2;
    }
    divide(){
        return this.num1 / this.num2;
    }
}
let calc = new Calculator(10, 5);
console.log(calc.add());
console.log(calc.subtract());
console.log(calc.multiply());
console.log(calc.divide());
```

Output:



```
[Running] node "c:\Users\vishn\OneDrive\Desktop\My Repository\Javascript\Assign.js"
15
5
50
2

[Done] exited with code=0 in 0.18 seconds
```

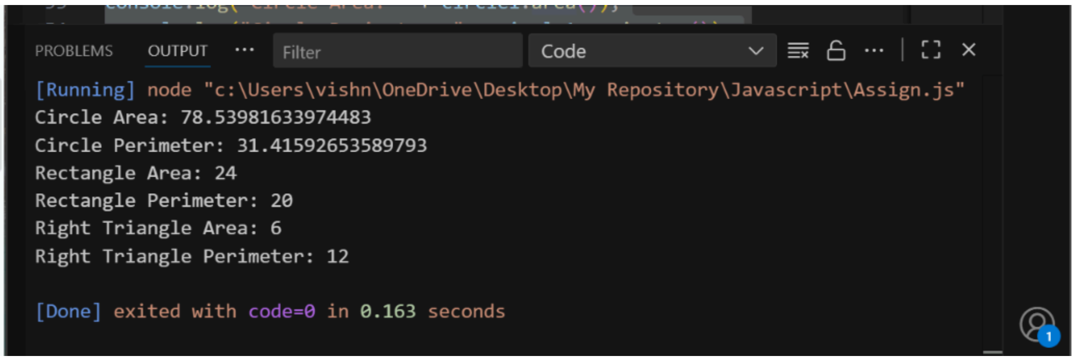
Problem Statement 6: Design a Class for a Geometric Shape Design a class called Shape that can represent various geometric shapes. Implement subclasses for specific shapes like Circle, Rectangle, and RightTriangle. Each subclass should have properties of that shape (e.g.. Width, height for a rectangle) and methods for calculating the area and perimeter of the shape. Create instances of these shapes and calculate their areas and perimeters.

Code:

```
class shape{
    constructor(){}
    area(){
        return 0;
    }
    perimeter(){
        return 0;
     }
}

class circle extends shape{
    constructor(radius){
        super();
        this.radius = radius;
    }
    area(){
        return Math.PI * this.radius * this.radius;
    }
    perimeter(){
        return 2 * Math.PI * this.radius;
    }
}

class rectangle extends shape{
    constructor(width, height){
        super();
        this.width = width;
        this.height = height;
    }
    area(){
        return this.width * this.height;
    }
    perimeter(){
        return 2 * (this.width + this.height);
    }
}

class rightTriangle extends shape{
    constructor(base, height){
        super();
        this.base = base;
        this.height = height;
    }
    area(){
        return 0.5 * this.base * this.height;
    }
    perimeter(){
        return this.base + this.height + Math.sqrt(this.base ** 2 +
this.height ** 2);
    }
}

let circle1 = new circle(5);
console.log("Circle Area: " + circle1.area());
console.log("Circle Perimeter: " + circle1.perimeter());
let rectangle1 = new rectangle(4, 6);
console.log("Rectangle Area: " + rectangle1.area());
console.log("Rectangle Perimeter: " + rectangle1.perimeter());
let rightTriangle1 = new rightTriangle(3, 4);
```

```
console.log("Right Triangle Area: " + rightTriangle1.area());
console.log("Right Triangle Perimeter: " + rightTriangle1.perimeter());
```

Output:



```
[Running] node "c:\Users\vishn\OneDrive\Desktop\My Repository\Javascript\Assign.js"
Circle Area: 78.53981633974483
Circle Perimeter: 31.41592653589793
Rectangle Area: 24
Rectangle Perimeter: 20
Right Triangle Area: 6
Right Triangle Perimeter: 12

[Done] exited with code=0 in 0.163 seconds
```

Problem Statement 7: Implement a Library System Create a class called Library that can manage a collection of books. Each book should be represented as an instance of a Book class (with properties like title, author, and availability). Implement methods in the Library class to add books, check out books, and return books. Keep track of the books available.

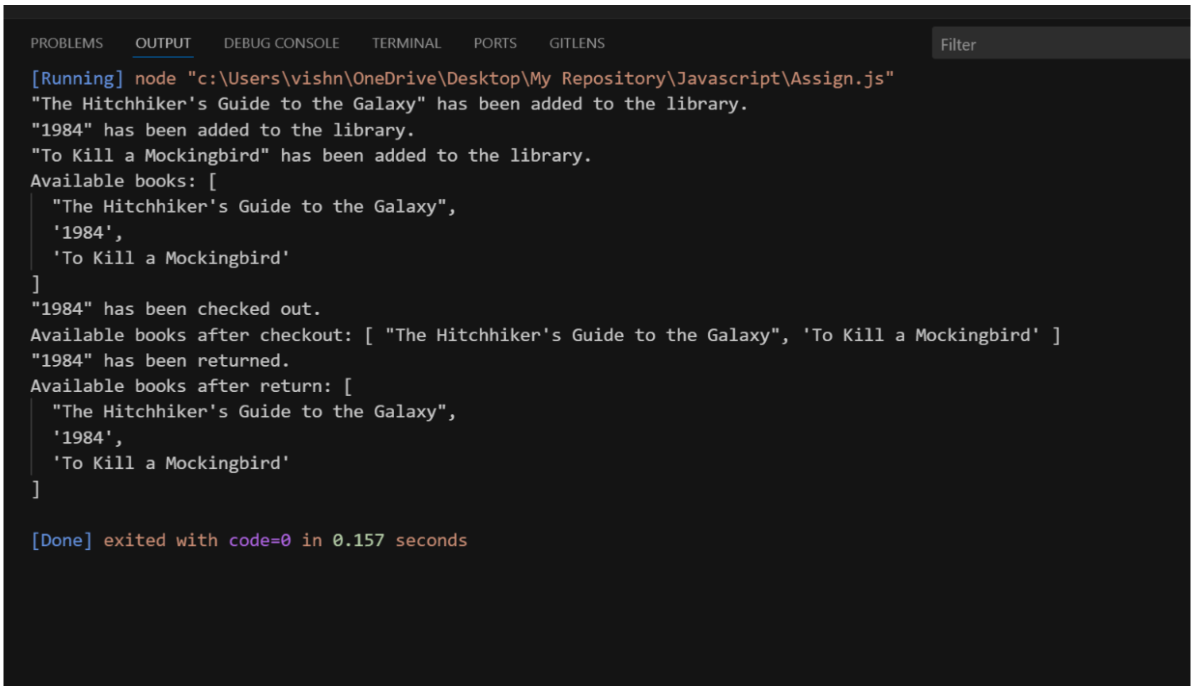
Code:

```
class Book {
  constructor(title, author, availability = true) {
    this.title = title;
    this.author = author;
    this.availability = availability;
  }
}

class Library {
  constructor() {
    this.books = [];
  }

  addBook(book) {
    this.books.push(book);
    console.log(`"${book.title}" has been added to the library.`);
  }

  checkOutBook(title) {
    const book = this.books.find(book => book.title === title &&
book.availability);
```

```
    if (book) {
      book.availability = false;
      console.log(`"${book.title}" has been checked out.`);
      return book;
    }
    console.log(`"${title}" is not available for checkout.`);
    return null;
  }

  returnBook(title) {
    const book = this.books.find(book => book.title === title &&
!book.availability);
    if (book) {
      book.availability = true;
      console.log(`"${book.title}" has been returned.`);
      return book;
    }
    console.log(`"${title}" was not checked out or does not exist.`);
    return null;
  }

  getAvailableBooks() {
    return this.books.filter(book => book.availability);
  }
}

// Example usage:
const library = new Library();

const book1 = new Book('The Hitchhiker\'s Guide to the Galaxy',
'Douglas Adams');
const book2 = new Book('1984', 'George Orwell');
const book3 = new Book('To Kill a Mockingbird', 'Harper Lee');

library.addBook(book1);
library.addBook(book2);
library.addBook(book3);

console.log('Available books:', library.getAvailableBooks().map(book =>
book.title));

library.checkOutBook('1984');
```

```
console.log('Available books after checkout:',
library.getAvailableBooks().map(book => book.title));

library.returnBook('1984');
console.log('Available books after return:',
library.getAvailableBooks().map(book => book.title));
```

Output:



```
[Running] node "c:\Users\vishn\OneDrive\Desktop\My Repository\Javascript\Assign.js"
"The Hitchhiker's Guide to the Galaxy" has been added to the library.
"1984" has been added to the library.
"To Kill a Mockingbird" has been added to the library.
Available books: [
  "The Hitchhiker's Guide to the Galaxy",
  '1984',
  'To Kill a Mockingbird'
]
"1984" has been checked out.
Available books after checkout: [ "The Hitchhiker's Guide to the Galaxy", 'To Kill a Mockingbird' ]
"1984" has been returned.
Available books after return: [
  "The Hitchhiker's Guide to the Galaxy",
  '1984',
  'To Kill a Mockingbird'
]

[Done] exited with code=0 in 0.157 seconds
```