NAME : Sri Nithyasri

REG.NO : 717823T153

DEPT : Electronics and TeleCommunication Engineering

MERN STACK TASK-week 2&3(Q1-Q35)

TASK 1.1: Implement a function to calculate the factorial of a number using recursion.

Program :

```
<!DOCTYPE html>
<html>
   <head>
      <title>
         SRI NITHYASRI-717823T153
      </title>
   </head>
   <body>
   <script>
    //factorial
    function factorial(num){

      if(num==0||num==1){ return 1;
      }else
      return num*factorial(num-1);
    }
    //var num=prompt("Enter any number: ");
    console.log("factorial  is "+ factorial(0));
    console.log("factorial is "+ factorial(1));
    console.log("factorial  is "+ factorial(6));
   </script>
   </body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
 factorial  is 1
 factorial is 1
 factorial  is 720
```

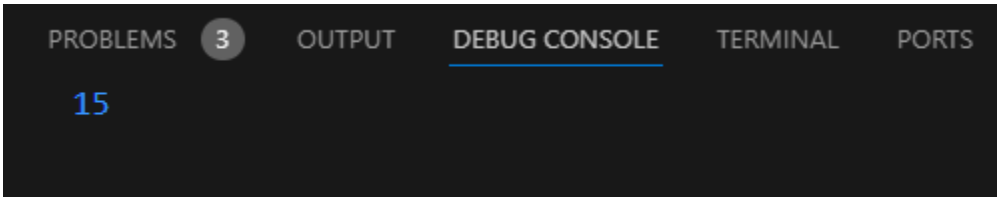
TASK 1 .2: Write a recursive function to find the nth Fibonacci number

Program :

```
<!DOCTYPE html>
<html>
   <head>
      <title>
         SRI NITHYASRI-717823T153
      </title>
   </head>
   <body>
      <script>
         function fibNum(num){ if(num==0 ||
            num==1){
               return 0;
            }else{
               return (num-2)+(num-1);
            }
         }
         console.log(fibNum(9));
      </script>
   </body>

</html>
```

Output :



TASK 1.3: Create a function to determine the total number of ways one can climb a staircase with 1, 2, or 3 steps at a time using recursion.

Program :

```
<!DOCTYPE html>
<html>
   <head>
      <title>
         SRI NITHYASRI-717823T153
      </title>
   </head>
   <body>
      <script>
        function countWays(n){
         if(n<0){
```

```
                return 0;
        }else if(n==0){ return 1;
                }else{
                return (n-1)+(n-2)+(n-3);
         }
         }
         console.log(countWays(8));
       </script>
    </body>

</html>
```

Output :

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

 18
```

TASK 1.4: Write a recursive function to flatten a nested array structure.

Program :

```
 <!DOCTYPE html>
<html>
   <head>
      <title>
         SRINITHYASRI-717823T153
      </title>
   </head>
   <body>
      <script>
       var arr=["apple","banana","mango"];
       arr.forEach(element=>console.log(element));
      </script>
   </body>

</html>
```

Ouput :

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

  apple
  banana
  mango
```

TASK1.5:Implement the recursive function of Tower Of Hanoi

Program :

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Sri Nithyasri-717823T153</title>
</head>
<body>
   <script>
    function towerOfHanoi(n, source, auxiliary, target) { if (n
    === 1) {
       console.log(`Move disk 1 from ${source} to ${target}`); return;
   }
    towerOfHanoi(n - 1, source, target, auxiliary); console.log(`Move
   disk ${n} from ${source} to ${target}`) towerOfHanoi(n - 1, auxiliary,
   source, target);
  }
   const numberOfDisks = 3; towerOfHanoi(numberOfDisks,
  'A', 'B', 'C');
    </script>
</body>
 </html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  Move disk 1 from A to C
  Move disk 2 from A to B
  Move disk 1 from C to B
  Move disk 3 from A to C
  Move disk 1 from B to A
  Move disk 2 from B to C
  Move disk 1 from A to C
```

TASK2.1 :Write a function that takes arbitrary arguments as input and provides its sum as the result.

Program :

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   <script>
      function arbSum(){ var
         sum=0;
         for(var num of arguments){
            sum+=num;
         }
         return sum;
      }
      console.log(arbSum(23,56,89));
      console.log(arbSum(67,78,89,0));
      </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  168
  234
```

TASK2.2: Modify a function to accept an array of numbers and return their sum using the spread syntax

Program :

```
<!DOCTYPE html>
<html><head>
    <title>
        Sri Nithyasri-717823T153
    </title>
</head>
<body>
    <script>
     function sum(...nums){
  return nums.reduce((total,num)=>total+num,0);}
var nums = [5,6,8];
console.log(sum(...nums));
    </script>
</body>

</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
   19
```

TASK2.3: Create a deep clone of an object using JSON methods.

Program:

```
<!DOCTYPE html>
<html><head>
    <title>
        Sri Nithyasri-717823T153
    </title>
</head>
<body>
    <script>
      let object={
        name:"Sri Nithyasri",
        age:"18",
      }
      var str=JSON.stringify(object);
      console.log(str);
      var obj2=JSON.parse(str);
      console.log(obj2);
    </script>
</html>

</body>

</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  {"name":"Sri Nithyasri","age":"18"}
> {name: 'Sri Nithyasri', age: '18'}
```

TASK 2.4: Write a function that returns a new object, merging two provided objects using the spread syntax.

Program :

```
<!DOCTYPE html>
<html><head>
    <title>
        Sri Nithyasri-717823T153
    </title>
</head>
<body>
    <script>
      var obj1={
        name:"Sri",
        age:"18",
     };
      var obj2={
        name1:"Nithyasri",
        age1:"19",
     };
      var result={...obj1,...obj2};
      console.log(result);
    </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

> {name: 'Sri', age: '18', name1: 'Nithyasri', age1: '19'}
```

TASK2.5: Serialize a JavaScript object into a JSON string and then parse it back into an object.

Program :

\

```
<!DOCTYPE html>
<html><head>
    <title>
        Sri Nithyasri-717823T153
    </title>
</head>
<body>
    <script>
      var obj={
        name:"sri",
        age:"19",
      }
      var a=JSON.stringify(obj);
      console.log(a);
      var b=JSON.parse(a);
      console.log(b);
    </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  {"name":"sri","age":"19"}
> {name: 'sri', age: '19'}
```

TASK3.1: Create a function that returns another function, capturing a local variable

Program :

```
<!DOCTYPE html>
<html><head>
    <title>
        Sri Nithyasri-717823T153
    </title>
</head>
<body>
    <script>
  function sqrFun(){
return function(num) {
  return num*num;
};
}
const sqr = sqrFun();
console.log(sqr(2));
console.log(sqr(89));
console.log(sqr(90));
    </script>
</body>
</html>
```

Output :

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
4
7921
8100
```

TASK3.2: Implement a basic counter function using closure, allowing incrementing and displaying the current count.

Program :

```
    <!DOCTYPE html>
<html><head>
    <title>
        Sri Nithyasri-717823T153
    </title>
</head>
<body>
    <script>
        function createCounter(){
    let a=0;
    if(a==0){
        console.log("Count Variable is Created");
    }
    return function(){
      a++;
      console.log(`The Current count is : ${a}`);
    };
  }
  const counter=createCounter();
  counter();
  counter();
  counter();
    </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  Count Variable is Created
  The Current count is : 1
  The Current count is : 2
  The Current count is : 3
```

TASK3.4: Use closures to create private variables within a function

Program :

```
<!DOCTYPE html>
<html><head>
    <title>
        Sri Nithyasri-717823T153
    </title>
</head>
<body>
    <script>
       function Count(){
            var a=1;
            return {
                increment:function(){
                    a++;
                    return a; },
                decrement:function(){
                    a--;
                    return a; },
                getCount:function(){
                    return a; }}
        }
        let count=Count();
        console.log(count.increment());
        console.log(count.decrement());
        console.log(count.getCount());
        console.log(count.c);
     </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  2
  1
  1
  undefined
```

TASK3.3: Write a function to create multiple counters, each with its own separate count.

Program :

```
<!DOCTYPE html>
<html>
<title>
Sri Nithyasri-717823T153
</title>
<body>
<script>
function createCounter() {
let count = 0;
return {
increment: function() {
count++;
},
getCount: function() {
return count;
}
};
}
const counter1 = createCounter();
const counter2 = createCounter();
counter1.increment();
counter1.increment();
console.log(counter1.getCount());

counter2.increment();
console.log(counter2.getCount());
</script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
  2
  1
```

TASK3.5: Build a function factory that generates functions based on some input using closures.

Program :

```
<!DOCTYPE html>
 <html>
    <title>Sri Nithyasri-717823T153</title>
    <body>
        <script>
           function multiplier(factor) {
    return function(number) {
        return number * factor;
    };
}
const double = multiplier(2);
const triple = multiplier(3);
console.log(double(9));
console.log(triple(8));
        </script>
    </body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  18
  24
```

TASK4.1: Create a new promise that resolves after a set number of seconds and returns a greeting.

Program :

```
 <!DOCTYPE html>
<html><head>
    <title>
        Sri Nithyasri-717823T153
    </title>
</head>
<body>
    <script>
       function myPromise(){
        return new Promise((resolve)=>{
            setTimeout(()=>{
                resolve();
               console.log("Warm Greetings(this text takes 5000milli seconds to load)");
```

```
            },5000); })
 }
      myPromise();
     </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
  Warm Greetings(this text takes 5000milli seconds to load)
```

TASK4.2: Fetch data from an API using promises, and then chain another promise to process this data.

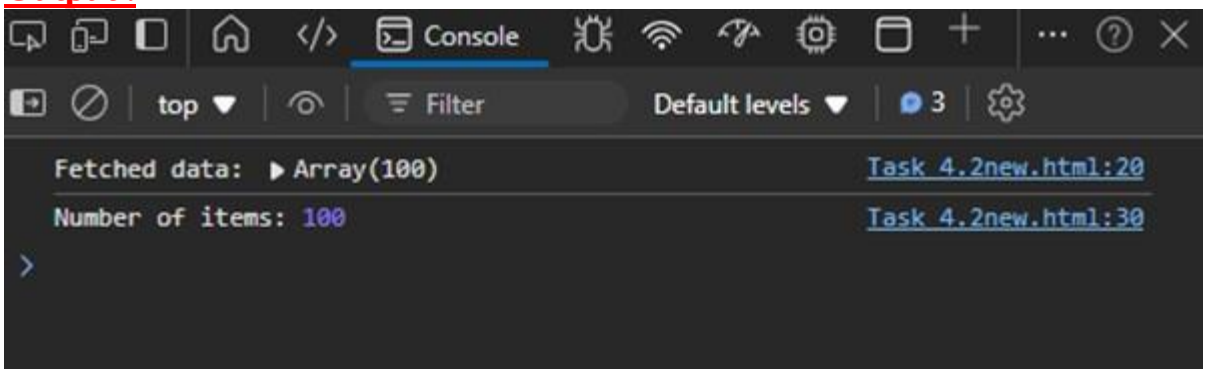
Program :

```
<html>
   <head>
<title>Sri Nithyasri-717823T153</title>
</head>
<body>
<script>
function fetchData(url) {
return fetch(url)
.then(response => response.json())
.then(data => {
console.log('Fetched data:', data);
return data;
})
.then(data => {
const count = data.length;
console.log('Number of items:', count);

})
.catch(error => {
console.log('Error:', error);
});
}
const apiUrl = 'https://jsonplaceholder.typicode.com/posts';
fetchData(apiUrl);
</script>
</body>
```

</html>

Output :



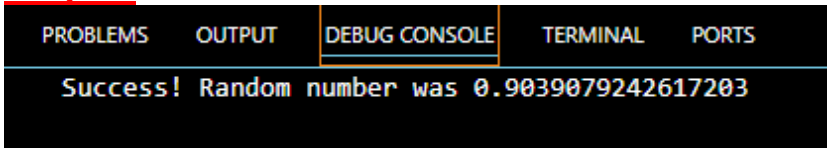
TASK4.3: Create a promise that either resolves or rejects based on a random number.

Program :

```
<!DOCTYPE html>
<html>
<head>
    <title>Sri Nithyasri-717823T153</title>
</head>
<body>
    <script>
  const randomPromise = new Promise((resolve, reject) => {
    const randomNumber = Math.random();
    if (randomNumber > 0.5) {
        resolve(`Success! Random number was ${randomNumber}`);
    } else {
        reject(`Failure! Random number was ${randomNumber}`);
    }
});

randomPromise
    .then(result => console.log(result))
    .catch(error => console.error(error));
    </script>
</body>
</html>
```

Output :



TASK4.4: Use Promise.all to fetch multiple resources in parallel from an API.

Program :

```
<!DOCTYPE html>
<html>
<head>
    <title>Sri Nithyasri-717823T153</title>
</head>
<body>
    <script>
  const urls = [
    'https://httpbin.org/get',
    'https://httpbin.org/get',
    'https://httpbin.org/get',
    'https://httpbin.org/get'
];
Promise.all(urls.map((url)=>fetch(url).then((response)=>response.json())))
    .then((jsons)=>{
        jsons.forEach((json)=>console.log(json));
    })
    .catch((error)=>console.error('error:',error));
    </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
> {args: {…}, headers: {…}, origin: '121.200.48.24', url: 'https://httpbin.org/get'}
> {args: {…}, headers: {…}, origin: '121.200.48.24', url: 'https://httpbin.org/get'}
> {args: {…}, headers: {…}, origin: '121.200.48.24', url: 'https://httpbin.org/get'}
> {args: {…}, headers: {…}, origin: '121.200.48.24', url: 'https://httpbin.org/get'}
```

TASK4.5: Chain multiple promises to perform a series of asynchronous actions in sequence.

Program :

```
<!DOCTYPE html>
<html>
  <title>TASK 4.5</title>
  <body>
    <script>
 function step1() {
  return new Promise((resolve) => {
    console.log("Step 1: Fetching user data...");
    setTimeout(() => resolve({ userId: 1, name: "Rupa" }), 1000);
  });
}
```

```
 function step2(user) {
return new Promise((resolve) => {
console.log("Step 2: Fetching user posts...");
    setTimeout(() => resolve([{ id: 1, title: "Post 1" }, { id: 2, title: "Post 2" }]), 1000);
  });
 }
 function step3(posts) {
  return new Promise((resolve) => {
    console.log("Step 3: Saving posts...");
    setTimeout(() => resolve(" saved successfully!"), 1000);
  });
 }
 step1()
  .then(user => {
    console.log("User data:", user);
    return step2(user);
  })
  .then(posts => {
    console.log("User's posts:", posts);
    return step3(posts);
  })
  .then(message => {
    console.log(message);
  })
  .catch(error => {
    console.error("Error:", error);
  });
     </script>
  </body>
 </html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
  Step 1: Fetching user data...
> User data: {userId: 1, name: 'Rupa'}
  Step 2: Fetching user posts...
> User's posts: (2) [{…}, {…}]
  Step 3: Saving posts...
   saved successfully!
```

TASK5.1: : Rewrite a promise-based function using async/await.

Program :

```
<!DOCTYPE html>
<html>
<head>
    <title>Sri Nithyasri-717823T153</title>
</head>
<body>
    <script>
        function PlaceFood(order){
            return new Promise((resolve)=>{
                setTimeout(()=>{
                    console.log(`${order} Order Placed.`);
                    resolve(order);
                },1000);    })  }
        function DeleiverFood(order){
            return new Promise((resolve)=>{
                setTimeout(()=>{
                    console.log(`${order} Order Delivered.`);
                    resolve(`${order} Order Delivered.`);
                },1000); }) }
        async function orders(food){
            const  orderss=await  PlaceFood(food);
            const deliver=await DeleiverFood(orderss);
            document.write(status); }
        orders("Biriyani");
    </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  Biriyani Order Placed.
  Biriyani Order Delivered.
```

TASK 5.2: Create an async function that fetches data from an API and processes it.

Program :

```
<!DOCTYPE html>
<html>
<head>
    <title>Sri Nithyasri-717823T153</title>
</head>
<body>
    <script>
        function PlaceFood(order){
            return new Promise((resolve)=>{
                setTimeout(()=>{
                    console.log(`${order} Order Placed.`);
                    resolve(order);
                },1000);
            })
        }
        function PrepareFood(order){
            return new Promise((resolve)=>{
                setTimeout(()=>{
                    console.log(`${order} Order Prepared.`);
                    resolve(order);
                },1000);
            })
        }
        function DeleiverFood(order){
            return new Promise((resolve)=>{
                setTimeout(()=>{
                    console.log(`${order} Order Delivered.`);
                    resolve(`${order} Order Delivered.`);
                },1000);
            })
        }
        async function orders(food){
            const orderss=await PlaceFood(food);
            const Prepare=await PrepareFood(orderss);
            const deliver=await DeleiverFood(Prepare);
            document.write(status);
        }
        orders("Naan and Paneer butter Masala");
        </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  Naan and Paneer butter Masala Order Placed.
  Naan and Paneer butter Masala Order Prepared.
  Naan and Paneer butter Masala Order Delivered.
```

TASK 5.3: Implement error handling in an async function using try/catch.

Program :

```
<!DOCTYPE html>
<html>
  <title>TASK 5.3</title>
  <body>
    <script>
async function fetchData() {
  throw new Error('URL is missing!');
}
async function main() {
  try {
    const data = await fetchData();
    console.log('Data fetched:', data);
  } catch (error) {
    console.error('Error occurred:', error.message);
  }
}
main();
      </script>
  </body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  Error occurred: URL is missing!
```

TASK 5.4: Use async/await in combination with Promise.all.

Program :

```
<!DOCTYPE html>
<html>
<head>
    <title>Sri Nithyasri-717823T153</title>
</head>
<body>
    <script>
  function one(){
    return new Promise((resolve,reject)=>{
        resolve("Hello!!!! "); });
};
function two(){
    return new Promise((resolve, reject)=>{
        resolve("Sriiiiii "); });
};
function three(){
    return new Promise((resolve, reject)=>{
        return setTimeout(()=>{
            resolve("Nithyasriiiiii");
        }, 2000); });
};
async function promiseExecution(){
    let promise = await Promise.all([one(),two(),three()]);
    console.log(promise);
};
promiseExecution();
        </script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
> (3) ['Hello!!!! ', 'Sriiiiii ', 'Nithyasriiiiii']
```

TASK 5.5: Create an async function that waits for multiple asynchronous operations to complete before proceeding.

Program :

```
<!DOCTYPE html>
<html>
  <title>TASK 5.5</title>
  <body>
    <script>
```

```
function asyncOperation(name, delay) {
  return new Promise(resolve => {
    setTimeout(() => {
      console.log(`${name} completed`);
      resolve(name);
    }, delay);
  });
}
async function main() {
  try {
    const results = await Promise.all([
      asyncOperation('Operation 1', 1000),
      asyncOperation('Operation 2', 2000)
    ]);
  } catch (error) {
    console.error('Error occurred:', error.message);
  }
}
main();
      </script>
  </body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  Operation 1 completed
  Operation 2 completed
```

TASK 6.1: Create a module that exports a function, a class, and a variable.

Program:

```
function greet(name) {
    return `Hello, ${name}!`;
}
class Car {
    constructor(make, model) {
        this.make = make;
        this.model = model;
    }
    getDetails() {
        return `${this.make} ${this.model}`;
    }
}
const appVersion = "2.0.0";
export { greet, Car, appVersion };
```

```
import { greet, Car, appVersion } from './myModule.js';
console.log(greet('Sri'));
const myCar = new Car('Benz', 'Model S');
console.log(myCar.getDetails());
console.log(`App version: ${appVersion}`);
```

Output :

```
Hello, Sri!
Benz Model S
App version: 2.0.0
```

TASK 6.2: Import the module in another JavaScript file and use the exported entities.

Program :

```
function greet(name) {
    return `Hello, ${name}!`;
}
class Car {
    constructor(make, model) {
        this.make = make;
        this.model = model;
    }
    getDetails() {
        return `${this.make} ${this.model}`;
    }
}
const appVersion = "2.0.0";
export { greet, Car, appVersion };

import { greet, Car, appVersion } from './myModule.js';
console.log(greet('Sri'));
const myCar = new Car('Benz', 'Model S');
console.log(myCar.getDetails());
console.log(`App version: ${appVersion}`);
```

Output :

```
Hello, Sri!
Benz Model S
App version: 2.0.0
```

TASK 6.3: Use named exports to export multiple functions from a module.

Program:

```
export function add(a, b) {
    return a + b;
  }
  export function subtract(a, b) {
    return a - b;
  }
  export function multiply(a, b) {
    return a * b;
  }
  export function divide(a, b) {
    if (b === 0) {
      return 'Error: Division by zero';
    }
    return a / b;
  }
```

TASK 6.4: Use named imports to import specific functions from a module.

Program:

```
import { add, subtract, multiply, divide } from './myModule.js';
console.log(add(10, 20));
console.log(subtract(90,80));
console.log(multiply(55,56));
console.log(divide(50,5));
console.log(divide(90, 0));
```

Output (3and 4) :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
  C:\Program Files\nodejs\node.exe .\main.js
  30
  10
  3080
  10
  Error: Division by zero
```

TASK 6.5: Use default export and import for a primary function of a module.

Program:

```
export default function calculate(a, b, operation) {
    switch (operation) {
```

```
      case 'add':
        return a + b;
      case 'subtract':
        return a - b;
      case 'multiply':
        return a * b;
      case 'divide':
        if (b === 0) {
          return 'Error: Division by zero';
        }
        return a / b;
      default:
        return 'Invalid operation';
    }
  }

import calculate from './myModule.js';
console.log(calculate(40, 50, 'add'));
console.log(calculate(96, 29, 'subtract'));
console.log(calculate(18, 39, 'multiply'));
console.log(calculate(180, 5, 'divide'));
console.log(calculate(20, 0, 'divide'));
console.log(calculate(10, 5, 'unknown'));
```

Output :

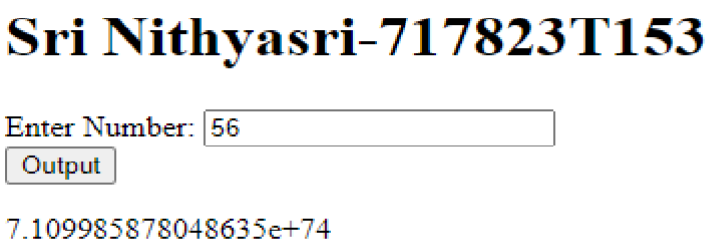
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
C:\Program Files\nodejs\node.exe .\main.js
90
67
702
36
Error: Division by zero
Invalid operation
```

TASK 7.1: Select an HTML element by its ID and change its content using JavaScript.

Program :

```
<!DOCTYPE html>
<html>
    <body>
        <h1>Sri Nithyasri-717823T153</h1>
        <form>
            <label>Enter Number:</label>
            <input type="number" id="num" name="numm"><br>
            <input type="button" id="cal" value="Output" onclick="fact()">
            <p id="numm"></p>
        </form>
    </body>
    <script>
function fact(){
    var num1=parseInt(document.getElementById("num").value);
    var res=factorial(num1);
    document.getElementById("numm").innerHTML=res;
}
function factorial(num){
    if(num==0) return 1;
    else
        return factorial(num-1)*num;
}
    </script>
</html>
```

Output :



TASK 7.2: Attach an event listener to a button, making it perform an action when clicked.

Program :

```
<!DOCTYPE html>
<html>
    <body>
        <h1>
            Sri Nithyasri-717823T153
        </h1>
        <form>
            <label>Enter Name:</label>
            <input type="text" id="nam" name="namm"><br>
            <input type="button" id="cal" value="Display" onclick="display()">
            <p id="num"></p>
        </form>
    </body>
    <script>
function display(){

    var name=document.getElementById("nam").value;
    document.getElementById("num").innerHTML=document.write(`Hello! ${name}`);
}
    </script>
</html>
```

Output :

TASK 7.3: : Create a new HTML element and append it to the DOM.

Program :

```
<!DOCTYPE html>
<html>
<body>
    <h1>
        Sri Nithyasri-717823T153
    </h1>
```

```
    <p>Adding new HTML Element</p>
    <div id="d">
        <p id="p1">This is the 1st line</p>
        <p id="p2">This is the 2nd line</p>
    </div>
<script>
const a=document.createElement("p");
const node=document.createTextNode("new line");
a.appendChild(node);
const ele=document.getElementById("d");
ele.appendChild(a);
</script>
</body>
</html>
```

Output :

TASK 7.4: Implement a function to toggle the visibility of an element.

Program :

```
<!DOCTYPE html>
<html lang="en">
<body>
        <p id="m">Hello<br></p>
        <button onclick="toggleElement()">
            Click to Toggle
        </button>
    <script>
        function toggleElement(){
            const a=document.getElementById('m');
            const vi=window.getComputedStyle(a).visibility;
            if (vi=='hidden')
                a.style.visibility='visible';
             else
                a.style.visibility='hidden';
        }
    </script>
</body>
```

```
</html>
```

Output :

Before toggling:

Sri Nithyasri-717823T153

Hello

Click to Toggle

After toggling:

Sri Nithyasri-717823T153

Click to Toggle

TASK 7.5: Use the DOM API to retrieve and modify the attributes of an element.

Program :

```
<!DOCTYPE html>
<html>
    <style>
        .attributee{
            color: blueviolet;
        };
    </style>
    <h1 id="Id">Hey Dood!</h1>
    <button onclick="addAttribute()">Add Element</button>
    <script>
        function addAttribute(){
            document.getElementById("Id").setAttribute("class","attributee");
        }
    </script>
</html>
```

Output :

Before adjusting the attribute:

Sri Nithyasri-717823T153

Welcome!!!!!!!!

Click to change color

After adjusting the attribute:

Sri Nithyasri-717823T153

Welcome!!!!!!!!

Click to change color