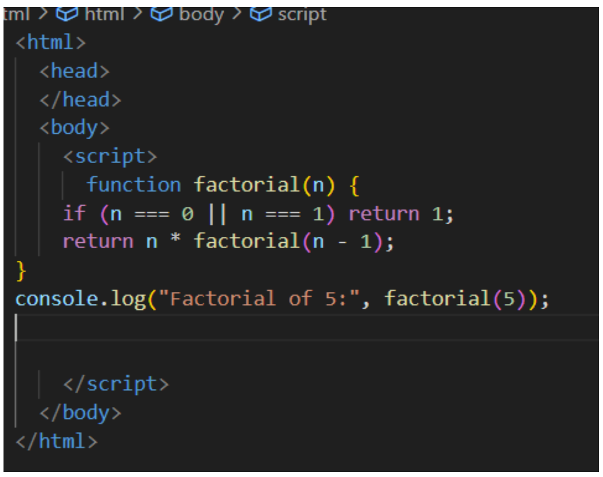
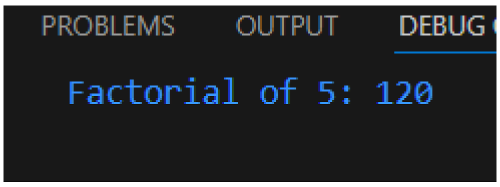
TASK 1:

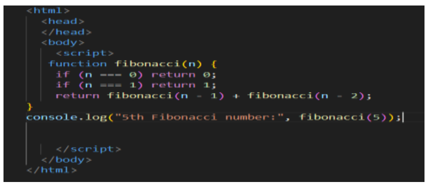


```
<html>
  <head>
  </head>
  <body>
    <script>
      function factorial(n) {
    if (n === 0 || n === 1) return 1;
    return n * factorial(n - 1);
}
console.log("Factorial of 5:", factorial(5));


    </script>
  </body>
</html>
```

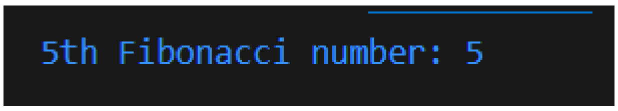


```
Factorial of 5: 120
```

TASK 2:

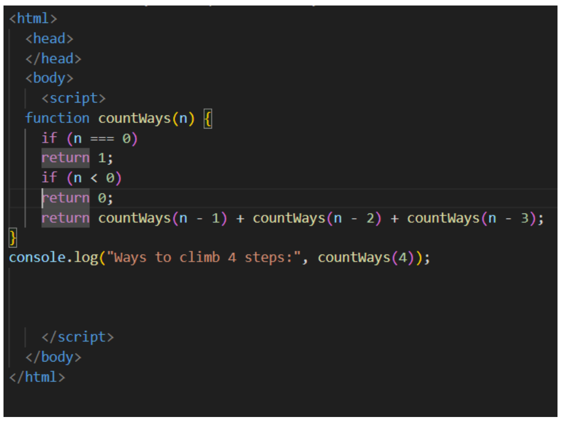


```
<html>
  <head>
  </head>
  <body>
    <script>
   function fibonacci(n) {
    if (n === 0) return 0;
    if (n === 1) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
console.log("5th Fibonacci number:", fibonacci(5));


    </script>
  </body>
</html>
```

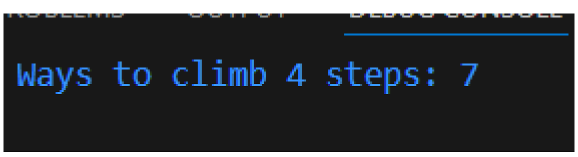


```
5th Fibonacci number: 5
```

TASK 3:

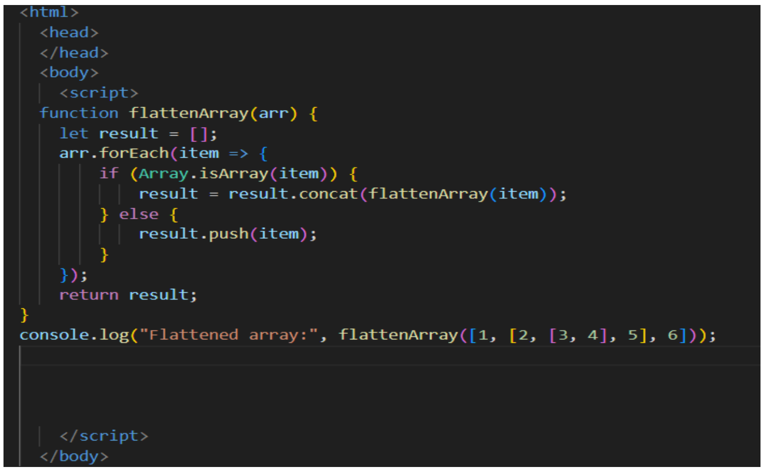


```
<html>
  <head>
  </head>
  <body>
    <script>
  function countWays(n) {
    if (n === 0)
    return 1;
    if (n < 0)
    return 0;
    return countWays(n - 1) + countWays(n - 2) + countWays(n - 3);
}
console.log("Ways to climb 4 steps:", countWays(4));


    </script>
  </body>
</html>
```

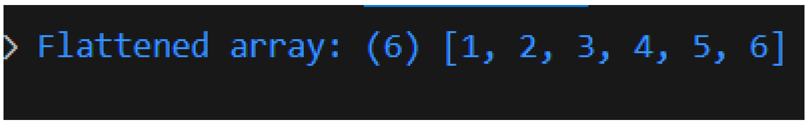


```
Ways to climb 4 steps: 7
```

TASK 4:

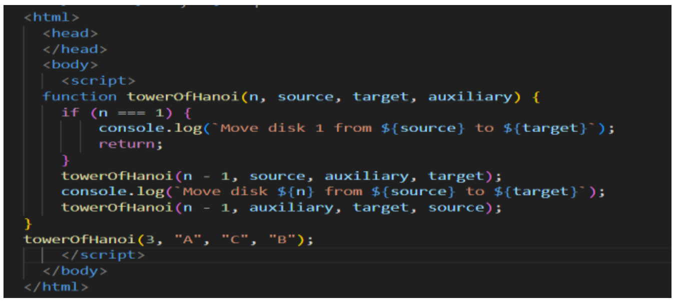


```
<html>
  <head>
  </head>
  <body>
    <script>
  function flattenArray(arr) {
    let result = [];
    arr.forEach(item => {
        if (Array.isArray(item)) {
            result = result.concat(flattenArray(item));
        } else {
            result.push(item);
        }
    });
    return result;
}
console.log("Flattened array:", flattenArray([1, [2, [3, 4], 5], 6]));



    </script>
  </body>
```

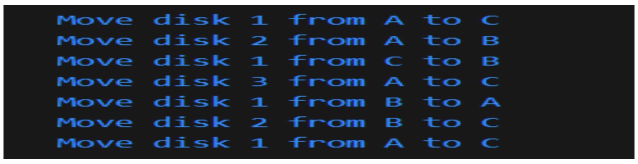


```
> Flattened array: (6) [1, 2, 3, 4, 5, 6]
```

TASK 5:

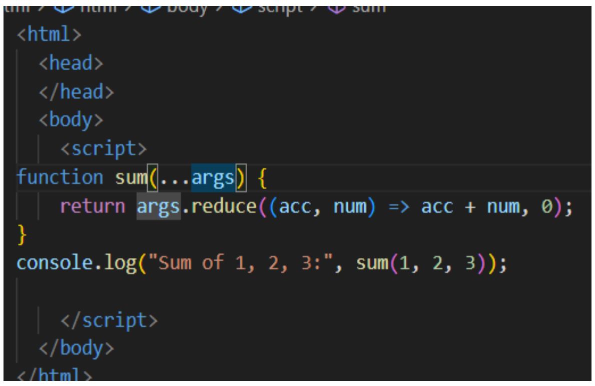


```
<html>
  <head>
  </head>
  <body>
    <script>
  function towerOfHanoi(n, source, target, auxiliary) {
    if (n === 1) {
        console.log(`Move disk 1 from ${source} to ${target}`);
        return;
    }
    towerOfHanoi(n - 1, source, auxiliary, target);
    console.log(`Move disk ${n} from ${source} to ${target}`);
    towerOfHanoi(n - 1, auxiliary, target, source);
}
towerOfHanoi(3, "A", "C", "B");
    </script>
  </body>
</html>
```

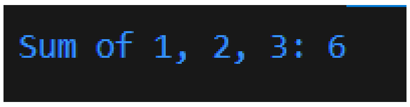


```
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

TASK 6:

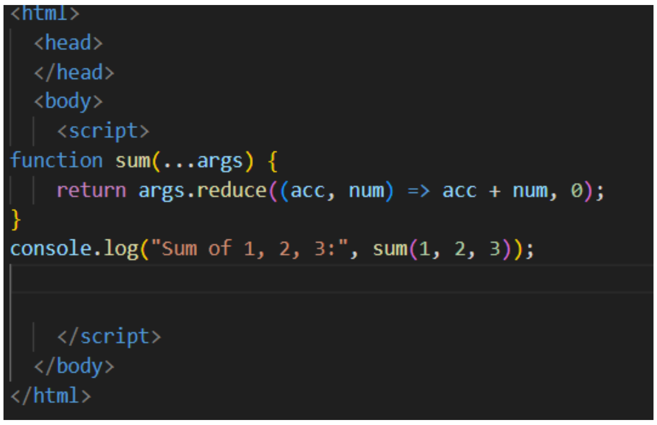


```
<html>
  <head>
  </head>
  <body>
    <script>
function sum(...args) {
    return args.reduce((acc, num) => acc + num, 0);
}
console.log("Sum of 1, 2, 3:", sum(1, 2, 3));

    </script>
  </body>
</html>
```

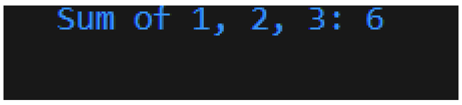


```
Sum of 1, 2, 3: 6
```

TASK 7:

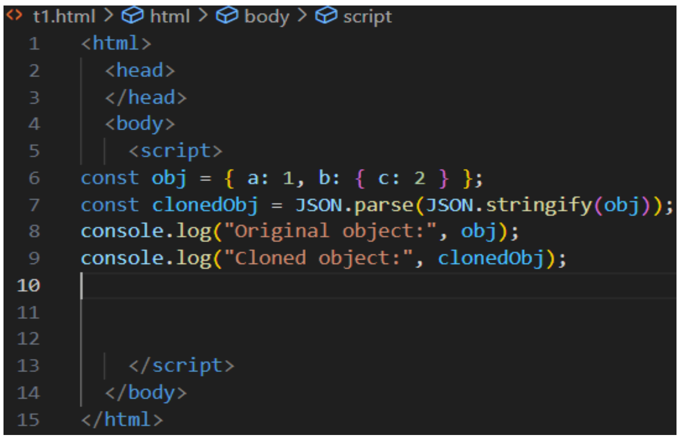


```
<html>
  <head>
  </head>
  <body>
    <script>
function sum(...args) {
    return args.reduce((acc, num) => acc + num, 0);
}
console.log("Sum of 1, 2, 3:", sum(1, 2, 3));


    </script>
  </body>
</html>
```



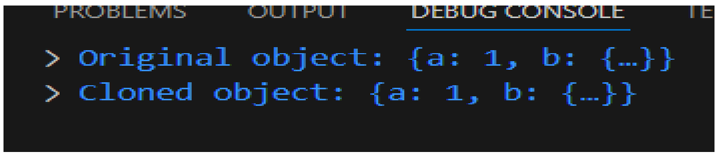
```
Sum of 1, 2, 3: 6
```

TASK 8:



t1.html > html > body > script

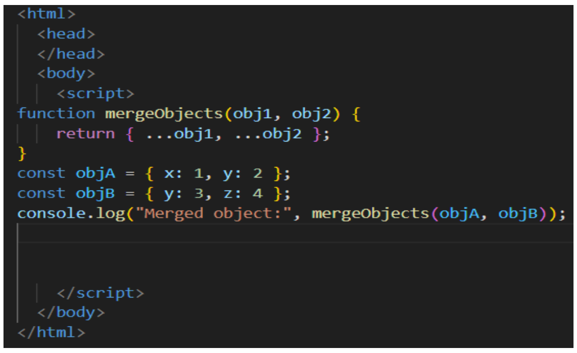
```
<html>
  <head>
  </head>
  <body>
    <script>
const obj = { a: 1, b: { c: 2 } };
const clonedObj = JSON.parse(JSON.stringify(obj));
console.log("Original object:", obj);
console.log("Cloned object:", clonedObj);



    </script>
  </body>
</html>
```



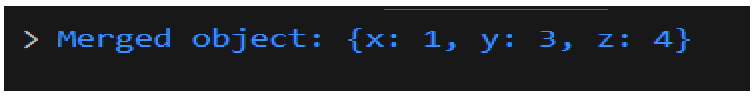
PROBLEMS OUTPUT DEBUG CONSOLE TE

```
> Original object: {a: 1, b: {…}}
> Cloned object: {a: 1, b: {…}}
```

TASK 9:

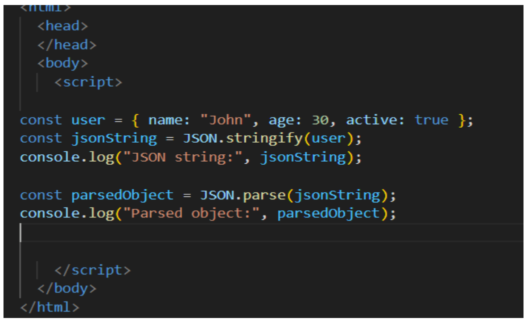


```
<html>
  <head>
  </head>
  <body>
    <script>
function mergeObjects(obj1, obj2) {
    return { ...obj1, ...obj2 };
}
const objA = { x: 1, y: 2 };
const objB = { y: 3, z: 4 };
console.log("Merged object:", mergeObjects(objA, objB));


    </script>
  </body>
</html>
```

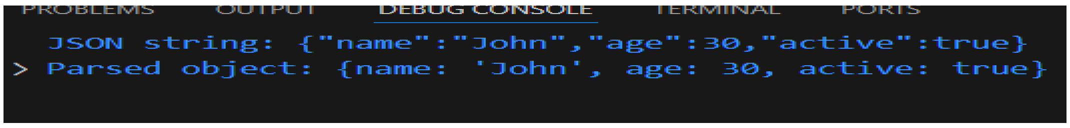


```
> Merged object: {x: 1, y: 3, z: 4}
```

TASK 10:

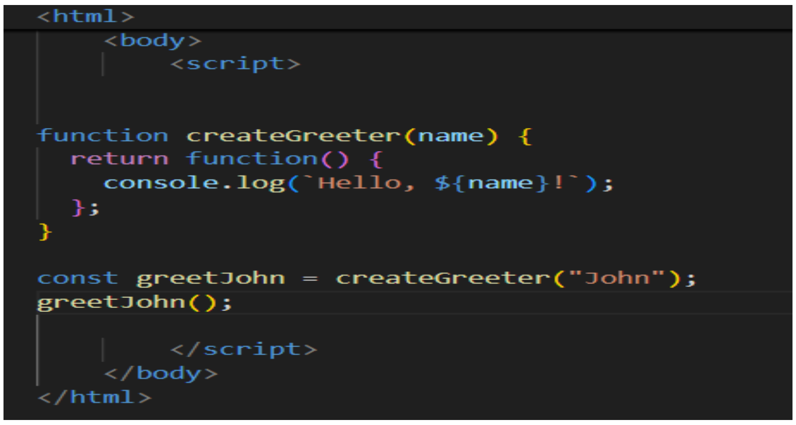


```
<html>
  <head>
  </head>
  <body>
    <script>

const user = { name: "John", age: 30, active: true };
const jsonString = JSON.stringify(user);
console.log("JSON string:", jsonString);

const parsedObject = JSON.parse(jsonString);
console.log("Parsed object:", parsedObject);


    </script>
  </body>
</html>
```

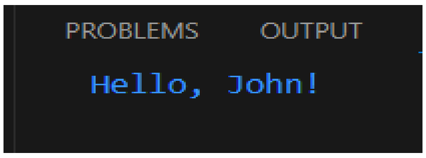


```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
  JSON string: {"name":"John","age":30,"active":true}
> Parsed object: {name: 'John', age: 30, active: true}
```

TASK 11:

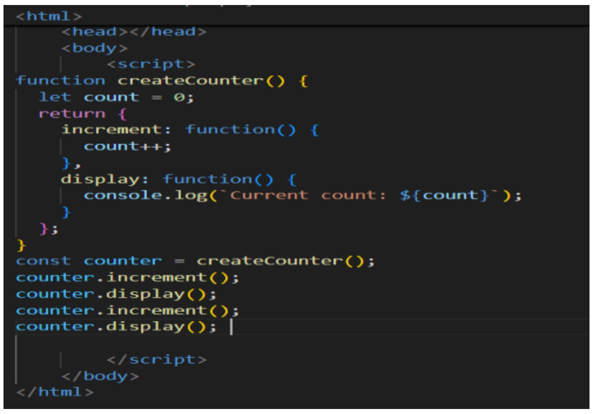


```
<html>
    <body>
        <script>

function createGreeter(name) {
  return function() {
    console.log(`Hello, ${name}!`);
  };
}

const greetJohn = createGreeter("John");
greetJohn();

        </script>
    </body>
</html>
```

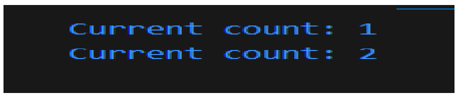


```
PROBLEMS    OUTPUT

  Hello, John!
```

TASK 12:

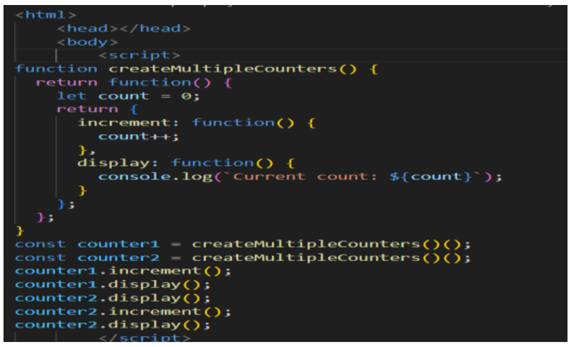


```
<html>
    <head></head>
    <body>
        <script>
function createCounter() {
  let count = 0;
  return {
    increment: function() {
      count++;
    },
    display: function() {
      console.log(`Current count: ${count}`);
    }
  };
}
const counter = createCounter();
counter.increment();
counter.display();
counter.increment();
counter.display();

        </script>
    </body>
</html>
```

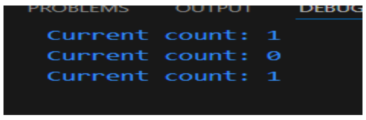


```
Current count: 1
Current count: 2
```

TASK 13:

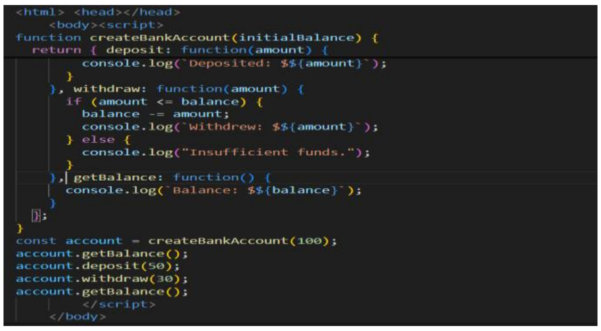


```
<html>
    <head></head>
    <body>
        <script>
function createMultipleCounters() {
  return function() {
    let count = 0;
    return {
      increment: function() {
        count++;
      },
      display: function() {
        console.log(`Current count: ${count}`);
      }
    };
  };
}
const counter1 = createMultipleCounters()();
const counter2 = createMultipleCounters()();
counter1.increment();
counter1.display();
counter2.display();
counter2.increment();
counter2.display();
        </script>
```

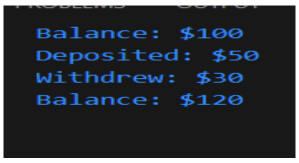


```
PROBLEMS    OUTPUT    DEBUG
  Current count: 1
  Current count: 0
  Current count: 1
```

TASK 14:

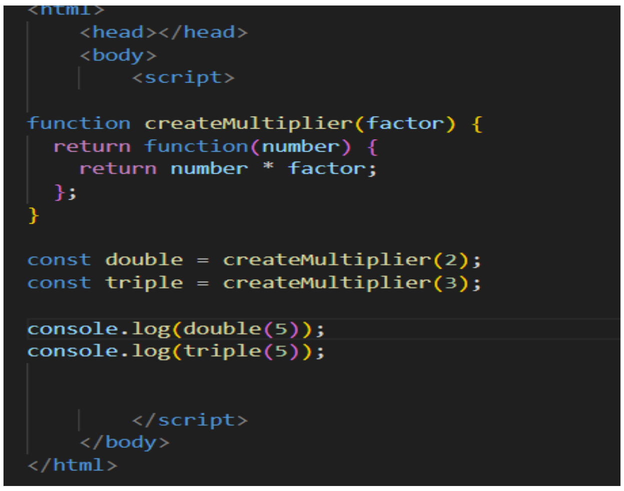


```
<html> <head></head>
    <body><script>
function createBankAccount(initialBalance) {
  return { deposit: function(amount) {
        console.log(`Deposited: $${amount}`);
      }
    }, withdraw: function(amount) {
      if (amount <= balance) {
        balance -= amount;
        console.log(`Withdrew: $${amount}`);
      } else {
        console.log("Insufficient funds.");
      }
    }, getBalance: function() {
      console.log(`Balance: $${balance}`);
    }
  };
}
const account = createBankAccount(100);
account.getBalance();
account.deposit(50);
account.withdraw(30);
account.getBalance();
        </script>
    </body>
```

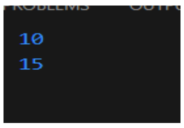


```
 Balance: $100
 Deposited: $50
 Withdrew: $30
 Balance: $120
```

TASK 15:

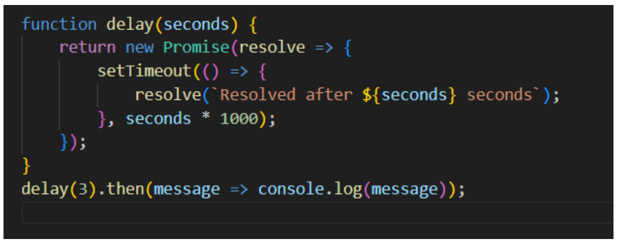


```
<html>
    <head></head>
    <body>
        <script>

function createMultiplier(factor) {
  return function(number) {
    return number * factor;
  };
}

const double = createMultiplier(2);
const triple = createMultiplier(3);

console.log(double(5));
console.log(triple(5));


        </script>
    </body>
</html>
```

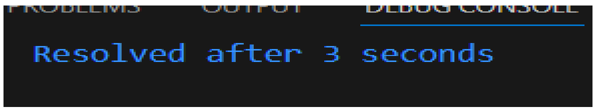


```
10
15
```

TASK 16:

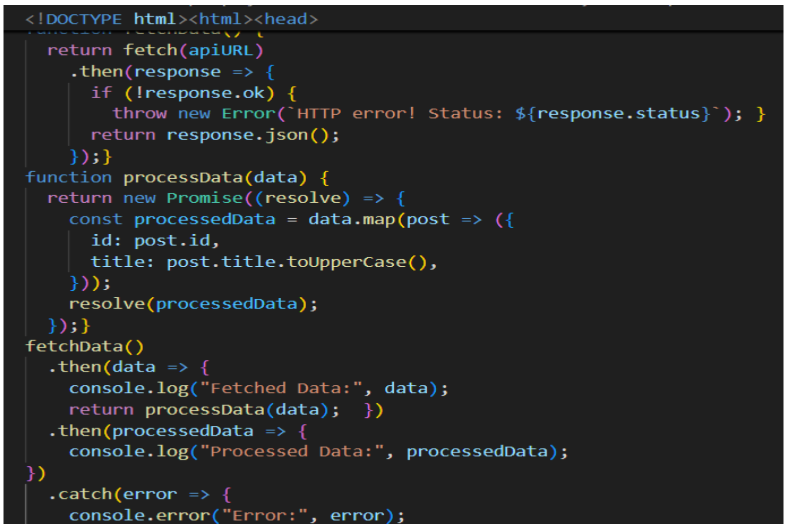


```
function delay(seconds) {
    return new Promise(resolve => {
        setTimeout(() => {
            resolve(`Resolved after ${seconds} seconds`);
        }, seconds * 1000);
    });
}
delay(3).then(message => console.log(message));
```

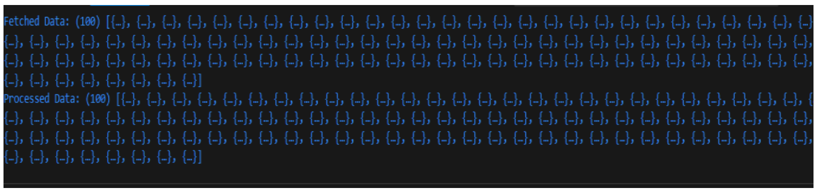


```
Resolved after 3 seconds
```

TASK 17:

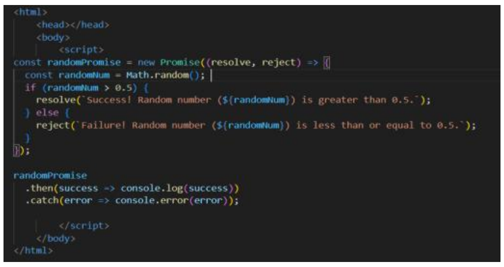


```
<!DOCTYPE html><html><head>
  return fetch(apiURL)
    .then(response => {
      if (!response.ok) {
        throw new Error(`HTTP error! Status: ${response.status}`); }
      return response.json();
    });}
function processData(data) {
  return new Promise((resolve) => {
    const processedData = data.map(post => ({
      id: post.id,
      title: post.title.toUpperCase(),
    }));
    resolve(processedData);
  });}
fetchData()
  .then(data => {
    console.log("Fetched Data:", data);
    return processData(data);  })
  .then(processedData => {
    console.log("Processed Data:", processedData);
})
  .catch(error => {
    console.error("Error:", error);
```

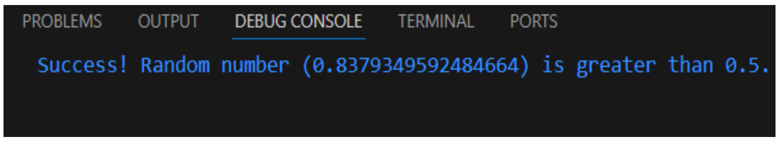


```
Fetched Data: (100) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
Processed Data: (100) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
```

TASK 18:

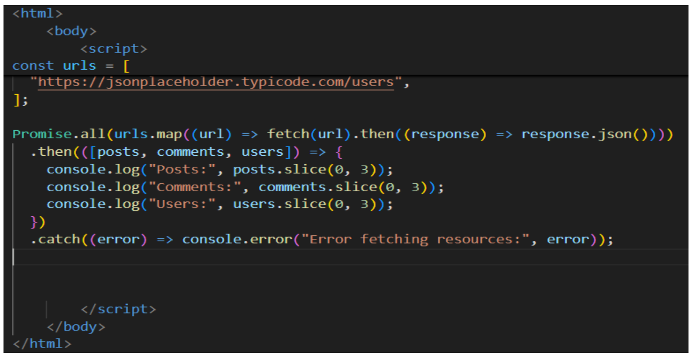


```
<html>
    <head></head>
    <body>
        <script>
const randomPromise = new Promise((resolve, reject) => {
  const randomNum = Math.random();
  if (randomNum > 0.5) {
    resolve(`Success! Random number (${randomNum}) is greater than 0.5.`);
  } else {
    reject(`Failure! Random number (${randomNum}) is less than or equal to 0.5.`);
  }
});

randomPromise
  .then(success => console.log(success))
  .catch(error => console.error(error));

        </script>
    </body>
</html>
```

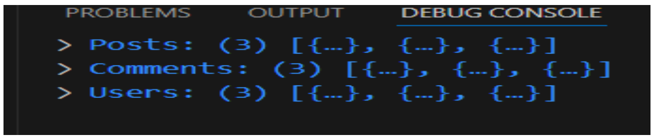


```
Success! Random number (0.8379349592484664) is greater than 0.5.
```

TASK 19:

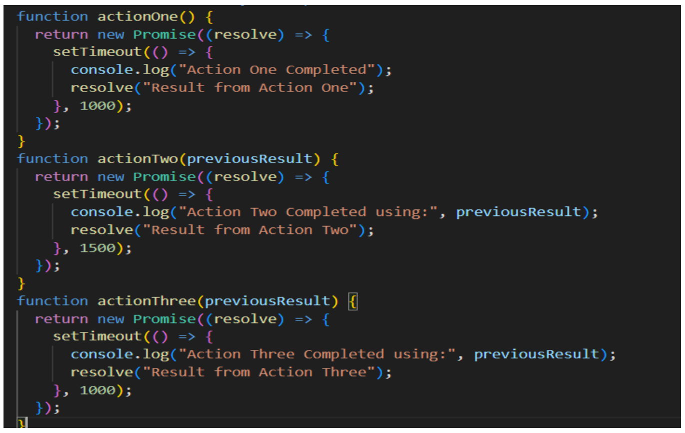


```
<html>
    <body>
        <script>
const urls = [
  "https://jsonplaceholder.typicode.com/users",
];

Promise.all(urls.map((url) => fetch(url).then((response) => response.json())))
  .then(([posts, comments, users]) => {
    console.log("Posts:", posts.slice(0, 3));
    console.log("Comments:", comments.slice(0, 3));
    console.log("Users:", users.slice(0, 3));
  })
  .catch((error) => console.error("Error fetching resources:", error));



        </script>
    </body>
</html>
```

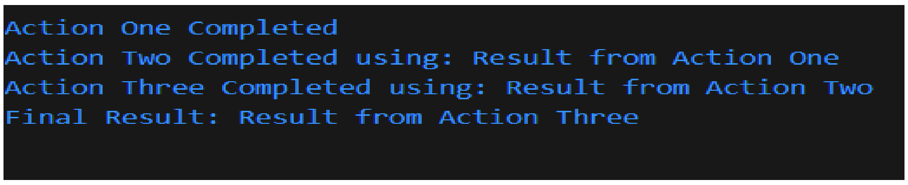


```
PROBLEMS    OUTPUT    DEBUG CONSOLE

  > Posts: (3) [{…}, {…}, {…}]
  > Comments: (3) [{…}, {…}, {…}]
  > Users: (3) [{…}, {…}, {…}]
```

TASK 20:

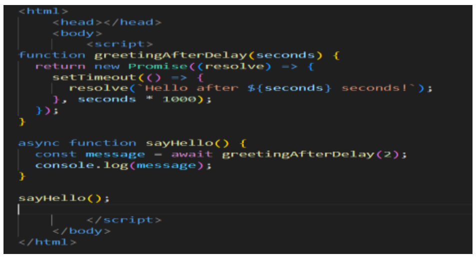


```
function actionOne() {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log("Action One Completed");
      resolve("Result from Action One");
    }, 1000);
  });
}
function actionTwo(previousResult) {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log("Action Two Completed using:", previousResult);
      resolve("Result from Action Two");
    }, 1500);
  });
}
function actionThree(previousResult) {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log("Action Three Completed using:", previousResult);
      resolve("Result from Action Three");
    }, 1000);
  });
}
```

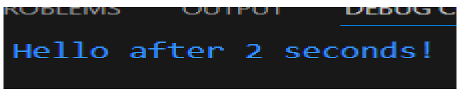


```
Action One Completed
Action Two Completed using: Result from Action One
Action Three Completed using: Result from Action Two
Final Result: Result from Action Three
```

TASK 21:

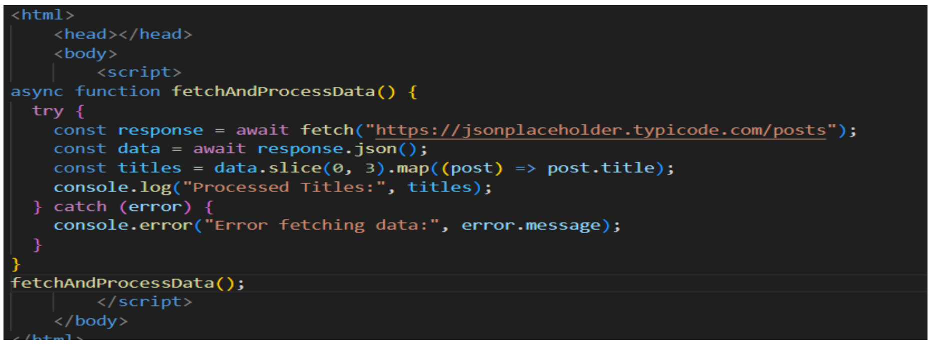


```
<html>
    <head></head>
    <body>
        <script>
function greetingAfterDelay(seconds) {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve(`Hello after ${seconds} seconds!`);
    }, seconds * 1000);
  });
}

async function sayHello() {
  const message = await greetingAfterDelay(2);
  console.log(message);
}

sayHello();

        </script>
    </body>
</html>
```

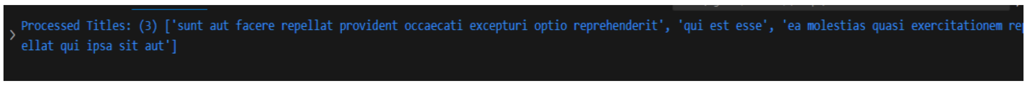


```
ROBLEMS    OUTPUT    DEBUG C

Hello after 2 seconds!
```

TASK 22:

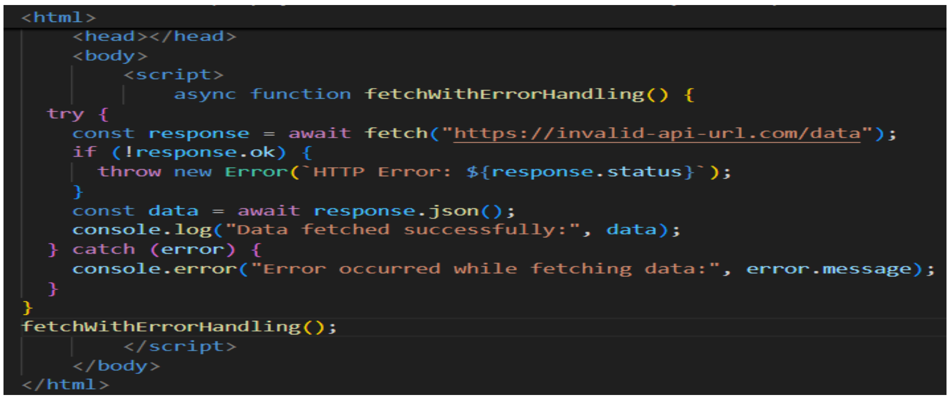


```
<html>
    <head></head>
    <body>
        <script>
async function fetchAndProcessData() {
  try {
    const response = await fetch("https://jsonplaceholder.typicode.com/posts");
    const data = await response.json();
    const titles = data.slice(0, 3).map((post) => post.title);
    console.log("Processed Titles:", titles);
  } catch (error) {
    console.error("Error fetching data:", error.message);
  }
}
fetchAndProcessData();
        </script>
    </body>
</html>
```

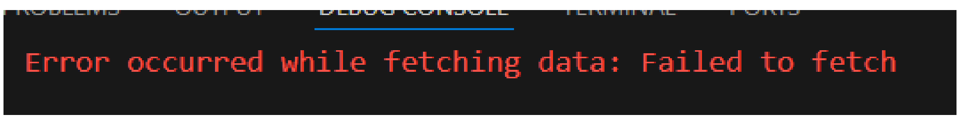


```
› Processed Titles: (3) ['sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'qui est esse', 'ea molestias quasi exercitationem re
  ellat qui ipsa sit aut']
```

TASK 23:

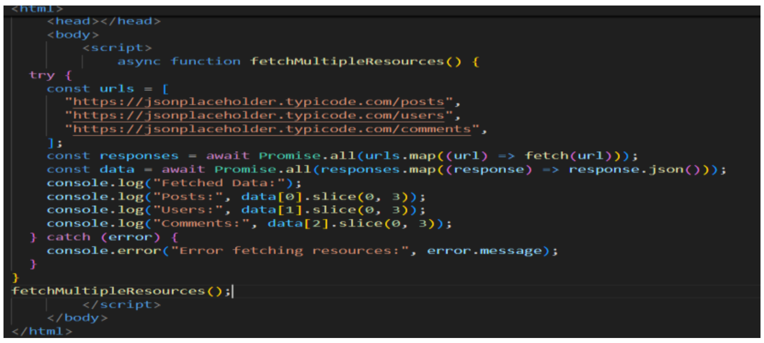


```
<html>
    <head></head>
    <body>
        <script>
            async function fetchWithErrorHandling() {
  try {
    const response = await fetch("https://invalid-api-url.com/data");
    if (!response.ok) {
      throw new Error(`HTTP Error: ${response.status}`);
    }
    const data = await response.json();
    console.log("Data fetched successfully:", data);
  } catch (error) {
    console.error("Error occurred while fetching data:", error.message);
  }
}
fetchWithErrorHandling();
        </script>
    </body>
</html>
```

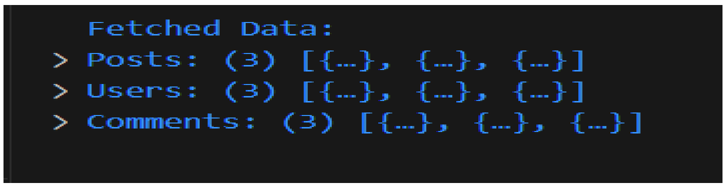


```
Error occurred while fetching data: Failed to fetch
```

TASK 24:

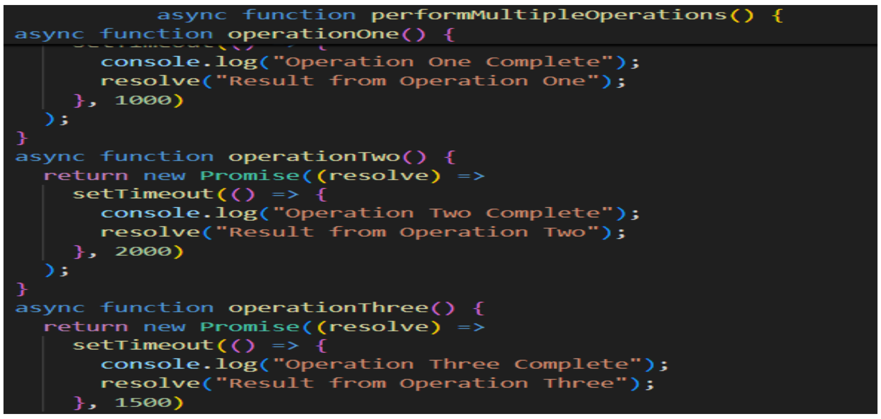


```
<html>
    <head></head>
    <body>
        <script>
            async function fetchMultipleResources() {
  try {
    const urls = [
      "https://jsonplaceholder.typicode.com/posts",
      "https://jsonplaceholder.typicode.com/users",
      "https://jsonplaceholder.typicode.com/comments",
    ];
    const responses = await Promise.all(urls.map((url) => fetch(url)));
    const data = await Promise.all(responses.map((response) => response.json()));
    console.log("Fetched Data:");
    console.log("Posts:", data[0].slice(0, 3));
    console.log("Users:", data[1].slice(0, 3));
    console.log("Comments:", data[2].slice(0, 3));
  } catch (error) {
    console.error("Error fetching resources:", error.message);
  }
}
fetchMultipleResources();
        </script>
    </body>
</html>
```

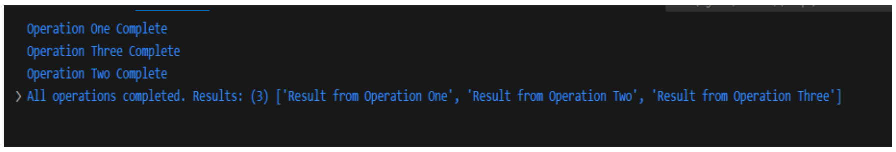


```
  Fetched Data:
> Posts: (3) [{…}, {…}, {…}]
> Users: (3) [{…}, {…}, {…}]
> Comments: (3) [{…}, {…}, {…}]
```

TASK 25:



```
          async function performMultipleOperations() {
async function operationOne() {
      console.log("Operation One Complete");
      resolve("Result from Operation One");
    }, 1000)
  );
}
async function operationTwo() {
  return new Promise((resolve) =>
    setTimeout(() => {
      console.log("Operation Two Complete");
      resolve("Result from Operation Two");
    }, 2000)
  );
}
async function operationThree() {
  return new Promise((resolve) =>
    setTimeout(() => {
      console.log("Operation Three Complete");
      resolve("Result from Operation Three");
    }, 1500)
```



```
  Operation One Complete
  Operation Three Complete
  Operation Two Complete
> All operations completed. Results: (3) ['Result from Operation One', 'Result from Operation Two', 'Result from Operation Three']
```