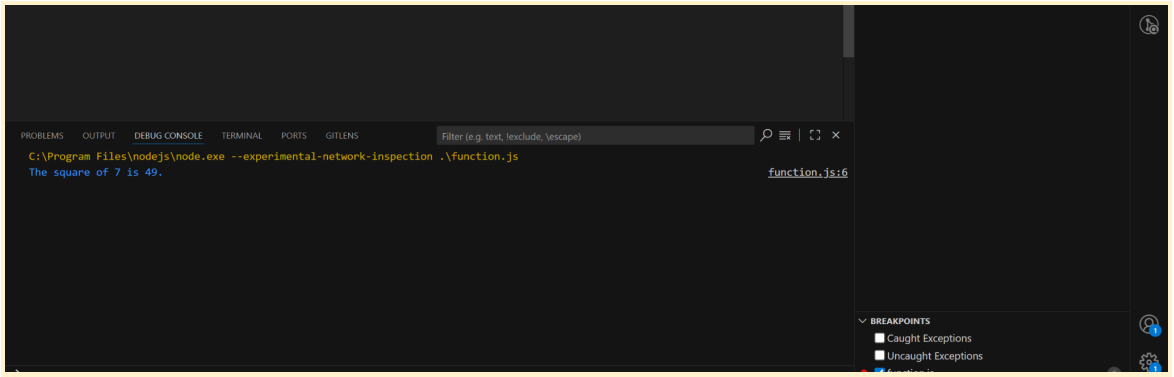
FUNCTION IN JAVASCRIPT - ANSWER

1. Create an arrow function called square that takes a number as an argument and returns its square. Use the arrow function to calculate the square of a given number and display the result.

   PROGRAM :

```
const square = (number) => {
    return number * number;
}
const mynumber = 7;
const result = square(mynumber);
console.log(`The square of ${mynumber} is ${result}.`);
```

   RESULT :



---

2. Create a JavaScript function called generateGreeting that takes a name as an argument and returns a personalized greeting message. Use this function to greet three different people.

   PROGRAM :

```
const generateGreeting = (name) => {
    return `Hello, ${name}! Welcome to the logical world.`;
}
// Person1
const person1 = 'Nikil';
const greeting1 = generateGreeting(person1);
```

```
console.log(greeting1);

// Person2
const person2 = 'Allen';
const greeting2 = generateGreeting(person2);
console.log(greeting2);

// Person3
const person3 = 'Prabavathi';
const greeting3 = generateGreeting(person3);
console.log(greeting3);
```

RESULT :

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS   Filter (e.g. text, !exclude, \escape)
C:\Program Files\nodejs\node.exe --experimental-network-inspection .\function.js
Hello, Nikil! Welcome to the logical world.                                   function.js:7
Hello, Allen! Welcome to the logical world.                                   function.js:12
Hello, Prabavathi! Welcome to the logical world.                              function.js:17
```

---

3. Create an IIFE (Immediately Invoked Function Expression) that calculates the square of a number and immediately displays the result.

PROGRAM :

```
(function (num) {
    const result = num * num;
    console.log(`The square of ${num} is ${result}`);
}
)
(7);
```

RESULT :

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS   Filter (e.g. text, !exclude, \escape)
C:\Program Files\nodejs\node.exe --experimental-network-inspection .\function.js
The square of 7 is 49                                                         function.js:3
```

---

4. Write a JavaScript function called calculateTax that takes an income as an argument and returns the amount of tax to be paid. Use a closure to handle different tax rates based on income ranges. Test the function with various incomes.

PROGRAM :

```
function calculateTax() {
    const taxBrackets = [
        { incomeThreshold: 0, rate: 0.10 },
        { incomeThreshold: 30000, rate: 0.15 },
        { incomeThreshold: 70000, rate: 0.20 },
        { incomeThreshold: 150000, rate: 0.25 },
        { incomeThreshold: 250000, rate: 0.30 }
    ];

    return function(income) {
        if (income < 0) {
            console.warn("Income cannot be negative. Returning 0
tax.");
            return 0;
        }

        let totalTax = 0;
        let remainingIncome = income;

        for (let i = 0; i < taxBrackets.length; i++) {
            const currentBracket = taxBrackets[i];
            const nextBracket = taxBrackets[i + 1];

            let taxableAmountInBracket;
            if (nextBracket) {
                taxableAmountInBracket =
Math.min(remainingIncome, nextBracket.incomeThreshold -
currentBracket.incomeThreshold);
            } else {
                taxableAmountInBracket = remainingIncome;
            }

            if (taxableAmountInBracket > 0) {
                totalTax += taxableAmountInBracket *
currentBracket.rate;
            }
```

```
            remainingIncome -= taxableAmountInBracket;

            if (remainingIncome <= 0) {
                break;
            }
        }
        return totalTax;
    };
}

const taxCalculator = calculateTax();

console.log("--- Tax Calculation Examples ---");

const incomesToTest = [0, 25000, 50000, 100000, 200000, 300000,
500000, -1000];

incomesToTest.forEach(income => {
    const taxPaid = taxCalculator(income);
    console.log(`Income: $${income.toLocaleString()} | Tax to be
paid: $${taxPaid.toFixed(2).toLocaleString()}`);
});

console.log("\n--- Breakdown for an income of $200,000 ---");
const incomeBreakdown = 200000;
let breakdownTax = 0;
console.log(`Income: $${incomeBreakdown.toLocaleString()}`);

const bracket1Amount = Math.min(incomeBreakdown, 30000);
const bracket1Tax = bracket1Amount * 0.10;
breakdownTax += bracket1Tax;
console.log(`  $${bracket1Amount.toLocaleString()} taxed at 10% =
$${bracket1Tax.toFixed(2).toLocaleString()}`);

const incomeAfterBracket1 = incomeBreakdown - bracket1Amount;
const bracket2Amount = Math.min(incomeAfterBracket1, 70000 -
30000);
const bracket2Tax = bracket2Amount * 0.15;
breakdownTax += bracket2Tax;
console.log(`  $${bracket2Amount.toLocaleString()} taxed at 15% =
$${bracket2Tax.toFixed(2).toLocaleString()}`);
```

```
const incomeAfterBracket2 = incomeAfterBracket1 - bracket2Amount;
const bracket3Amount = Math.min(incomeAfterBracket2, 150000 -
70000);
const bracket3Tax = bracket3Amount * 0.20;
breakdownTax += bracket3Tax;
console.log(`  $${bracket3Amount.toLocaleString()} taxed at 20% =
$${bracket3Tax.toFixed(2).toLocaleString()}`);

const incomeAfterBracket3 = incomeAfterBracket2 - bracket3Amount;
const bracket4Amount = Math.min(incomeAfterBracket3, 250000 -
150000);
const bracket4Tax = bracket4Amount * 0.25;
breakdownTax += bracket4Tax;
console.log(`  $${bracket4Amount.toLocaleString()} taxed at 25% =
$${bracket4Tax.toFixed(2).toLocaleString()}`);

const incomeAfterBracket4 = incomeAfterBracket3 - bracket4Amount;
const bracket5Amount = incomeAfterBracket4;
const bracket5Tax = bracket5Amount * 0.30;
breakdownTax += bracket5Tax;
if (bracket5Amount > 0) {
    console.log(`  $${bracket5Amount.toLocaleString()} taxed at
30% = $${bracket5Tax.toFixed(2).toLocaleString()}`);
}

console.log(`Total calculated tax:
$${breakdownTax.toFixed(2).toLocaleString()}`);
console.log(`Tax calculated by function:
$${taxCalculator(incomeBreakdown).toFixed(2).toLocaleString()}`);
```

RESULT :

```
--- Tax Calculation Examples ---                                   function.js:46
Income: $0 | Tax to be paid: $0.00                                 function.js:52
Income: $25,000 | Tax to be paid: $2500.00                         function.js:52
Income: $50,000 | Tax to be paid: $6000.00                         function.js:52
Income: $1,00,000 | Tax to be paid: $15000.00                      function.js:52
Income: $2,00,000 | Tax to be paid: $37500.00                      function.js:52
Income: $3,00,000 | Tax to be paid: $65000.00                      function.js:52
Income: $5,00,000 | Tax to be paid: $125000.00                     function.js:52
Income cannot be negative. Returning 0 tax.                        function.js:12
Income: $-1,000 | Tax to be paid: $0.00                            function.js:52
                                                                   function.js:55
--- Breakdown for an income of $200,000 ---
Income: $2,00,000                                                  function.js:58
  $30,000 taxed at 10% = $3000.00                                  function.js:63
  $40,000 taxed at 15% = $6000.00                                  function.js:69
  $80,000 taxed at 20% = $16000.00                                 function.js:75
  $50,000 taxed at 25% = $12500.00                                 function.js:81
Total calculated tax: $37500.00                                    function.js:91
Tax calculated by function: $37500.00                              function.js:92
>
Software_Development  master*  Launchpad  0 0  Launch Program (Software_Development)  Ln 1, Col 1 (3650 selected)  Spaces: 4
```

---

5. Write a JavaScript function called factorial that calculates the factorial of a non-negative integer using recursion. Test the function with different inputs.

PROGRAM :

```
function factorial(n) {
    if (n < 0) {
        console.error("Factorial is not defined for negative
numbers.");
        return NaN;
    }
    if (n === 0 || n === 1) {
        return 1;
    }
    return n * factorial(n - 1);
}

console.log("--- Factorial Calculation Examples ---");

const inputsToTest = [0, 1, 5, 7, 10, -3, 15];

inputsToTest.forEach(input => {
    const result = factorial(input);
    if (isNaN(result)) {
```

```
        console.log(`Factorial of ${input}: Invalid input
(negative number)`);
    } else {
        console.log(`Factorial of ${input}: ${result}`);
    }
});

console.log("\n--- Explanation of Factorial ---");
console.log("The factorial of a non-negative integer 'n', denoted
by n!, is the product of all positive integers less than or equal
to n.");
console.log("For example:");
console.log("5! = 5 * 4 * 3 * 2 * 1 = 120");
console.log("3! = 3 * 2 * 1 = 6");
console.log("0! = 1 (by definition)");
```

RESULT :

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS   Filter (e.g. text, !exclude, \escape)
C:\Program Files\nodejs\node.exe --experimental-network-inspection .\function.js
--- Factorial Calculation Examples ---                                   function.js:12
Factorial of 0: 1                                                        function.js:21
Factorial of 1: 1                                                        function.js:21
Factorial of 5: 120                                                      function.js:21
Factorial of 7: 5040                                                     function.js:21
Factorial of 10: 3628800                                                 function.js:21
Factorial is not defined for negative numbers.                            function.js:3
Factorial of -3: Invalid input (negative number)                         function.js:19
Factorial of 15: 1307674368000                                           function.js:21
                                                                         function.js:25
--- Explanation of Factorial ---
The factorial of a non-negative integer 'n', denoted by n!, is the product of all positive integers less than or equal t …nction.js:26
o n.
For example:                                                             function.js:27
5! = 5 * 4 * 3 * 2 * 1 = 120                                             function.js:28
3! = 3 * 2 * 1 = 6                                                       function.js:29
0! = 1 (by definition)                                                   function.js:30
```

---

6. Write a JavaScript function called curry that takes a function as an argument and returns a curried version of that function. The curried function should accept arguments one at a time and return a new function until all arguments are provided. Then, it should execute the original function with all arguments. Test the curry function with a function that adds two numbers.

PROGRAM :

```
function curry(func) {
    return function curried(...args) {
        if (args.length >= func.length) {
```

```
            return func(...args);
        } else {
            return function(...nextArgs) {
                return curried(...args, ...nextArgs);
            };
        }
    };
}

function add(a, b) {
    return a + b;
}

console.log("--- Original `add` function test ---");
console.log(`add(2, 3): ${add(2, 3)}`);

const curriedAdd = curry(add);

console.log("\n--- Curried `add` function test ---");

const addTwo = curriedAdd(2);
console.log(`curriedAdd(2)(3): ${addTwo(3)}`);

console.log(`curriedAdd(5, 10): ${curriedAdd(5, 10)}`);

const addTen = curriedAdd(10);
const addTenAndFive = addTen(5);
console.log(`curriedAdd(10)(5): ${addTenAndFive}`);

function multiply(x, y, z) {
    return x * y * z;
}

const curriedMultiply = curry(multiply);

console.log("\n--- Curried `multiply` function test ---");
console.log(`curriedMultiply(2)(3)(4):
${curriedMultiply(2)(3)(4)}`);
console.log(`curriedMultiply(2, 3)(4): ${curriedMultiply(2,
3)(4)}`);
console.log(`curriedMultiply(2)(3, 4): ${curriedMultiply(2)(3,
4)}`);
```

```
console.log(`curriedMultiply(2, 3, 4): ${curriedMultiply(2, 3,
4)}`);
```

RESULT :

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS   Filter (e.g. text, !exclude, \escape)

C:\Program Files\nodejs\node.exe --experimental-network-inspection .\function.js
--- Original `add` function test ---                                  function.js:17
add(2, 3): 5                                                          function.js:18
                                                                      function.js:22
--- Curried `add` function test ---
curriedAdd(2)(3): 5                                                   function.js:25
curriedAdd(5, 10): 15                                                 function.js:27
curriedAdd(10)(5): 15                                                 function.js:31
                                                                      function.js:39
--- Curried `multiply` function test ---
curriedMultiply(2)(3)(4): 24                                          function.js:40
curriedMultiply(2, 3)(4): 24                                          function.js:41
curriedMultiply(2)(3, 4): 24                                          function.js:42
curriedMultiply(2, 3, 4): 24                                          function.js:43
```