Practical 2

Task 1: Hoisting in Variables

Write a Node.js program that demonstrates variable hoisting using var, let, and const.

Print a variable before it is declared.

Show the difference between var, let, and const.

Explain the output.

JS Ravi_GF202348599_2.1.js > ...

```
1   console.log("=== Hoisting with var ===");
2   console.log(a);   // Access before declaration
3   var a = 10;
4   console.log(a);   // After assignment
5
6   console.log("\n=== Hoisting with let ===");
7   try {
8       console.log(b);   // Access before declaration
9   } catch (err) {
10      console.log("Error:", err.message);
11  }
12  let b = 20;
13  console.log(b);
14
15  console.log("\n=== Hoisting with const ===");
16  try {
17      console.log(c);   // Access before declaration
18  } catch (err) {
19      console.log("Error:", err.message);
20  }
21  const c = 30;
22  console.log(c);
23
```

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/node ./Ravi_GF202348599_2.1.js
=== Hoisting with var ===
undefined
10

=== Hoisting with let ===
Error: Cannot access 'b' before initialization
20

=== Hoisting with const ===
Error: Cannot access 'c' before initialization
30
```

Task 2: Function Declarations vs Expressions
Create two functions in Node.js:
A function declaration (function add(a,b) {})
A function expression (const multiply = function(a,b) {})
Call both functions before and after their definitions.

Ravi_GF202348599_2.2.js > ...

```
1  console.log("=== Function Declaration ===");
2  try {
3      console.log("Before definition:", add(2, 3)); // Works due to hoisting
4  } catch (err) {
5      console.log("Error before definition:", err.message);
6  }
7
8  function add(a, b) {
9      return a + b;
10 }
11
12 console.log("After definition:", add(5, 7)); // Works normally
13
14
15 console.log("\n=== Function Expression ===");
16 try {
17     console.log("Before definition:", multiply(2, 3)); // Fails
18 } catch (err) {
19     console.log("Error before definition:", err.message);
20 }
21
22 const multiply = function (a, b) {
23     return a * b;
24 };
25
26 console.log("After definition:", multiply(5, 7)); // Works after assignment
27
```

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/node ./Ravi_GF202348599_2.2.js
=== Function Declaration ===
Before definition: 5
After definition: 12

=== Function Expression ===
Error before definition: Cannot access 'multiply' before initialization
After definition: 35
```

Task 3: Arrow Functions vs Normal Functions
Create two functions inside an object:
One arrow function
One normal function
Both should print this.

```
const obj = {
    name: "MyObject",

    // Normal function
    normalFunc: function () {
        console.log("Normal Function this:", this);
    },

    // Arrow function
    arrowFunc: () => {
        console.log("Arrow Function this:", this);
    }
};

console.log("=== Calling methods ===");
obj.normalFunc(); // 'this' refers to obj
obj.arrowFunc();  // 'this' does not refer to obj
```

Output:

```
/usr/local/bin/node ./Ravi_GF202348599_2.3.js
=== Calling methods ===
> Normal Function this: {name: 'MyObject', normalFunc: f, arrowFunc: f}
> Arrow Function this: {}
```

Task 4: Higher Order Functions

Write a Node.js function calculate(operation, a, b) where operation is another function (like add, subtract).

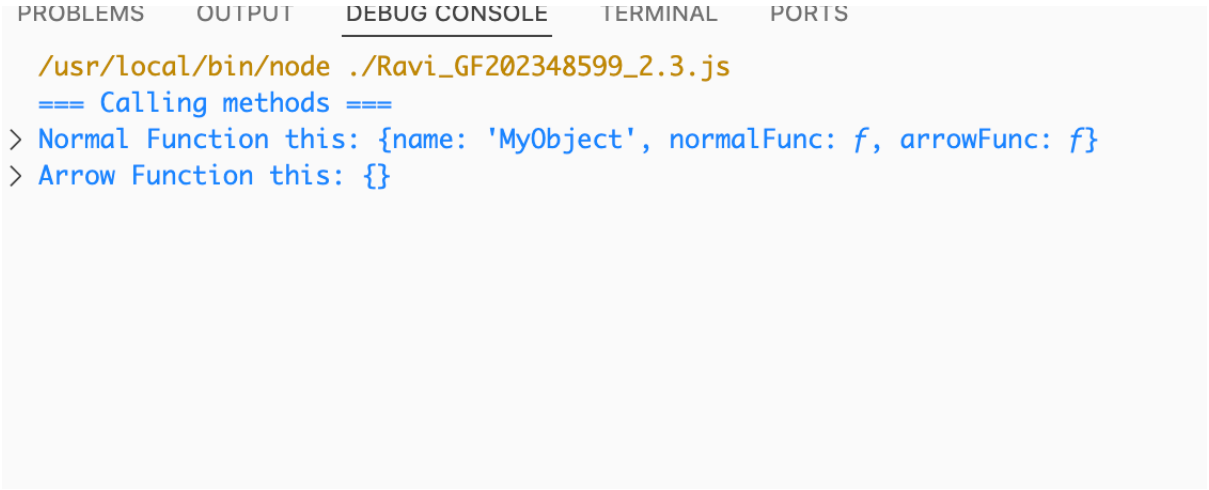
Pass different functions to calculate and print results.

Example: calculate((x,y) => x*y, 4, 5) should return 20.

Ravi_GF202348599_2.4.js > ...

```
// Higher Order Function
function calculate(operation, a, b) {
    return operation(a, b);
}

// Some operation functions
function add(x, y) {
    return x + y;
}

function subtract(x, y) {
    return x - y;
}

const multiply = (x, y) => x * y;
const divide = (x, y) => (y !== 0 ? x / y : "Cannot divide by zero");

// === Testing calculate ===
console.log("Addition:", calculate(add, 10, 5));          // 15
console.log("Subtraction:", calculate(subtract, 10, 5)); // 5
console.log("Multiplication:", calculate(multiply, 4, 5)); // 20
console.log("Division:", calculate(divide, 20, 4));        // 5
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
/usr/local/bin/node ./Ravi_GF202348599_2.3.js
=== Calling methods ===
> Normal Function this: {name: 'MyObject', normalFunc: f, arrowFunc: f}
> Arrow Function this: {}