**NAME: Sharvayu Zade**

**PRN: 23070521135**

**SEC: B(B2)**

**Practical 4 Final Task B2**

You are hired as a JavaScript developer for a Digital Locker System company. The company wants you to build a prototype program that secures user data (like PIN, secret notes), validates input, and provides some useful utility functions (like palindrome checks for secure PINs). The manager gave you the following requirements:

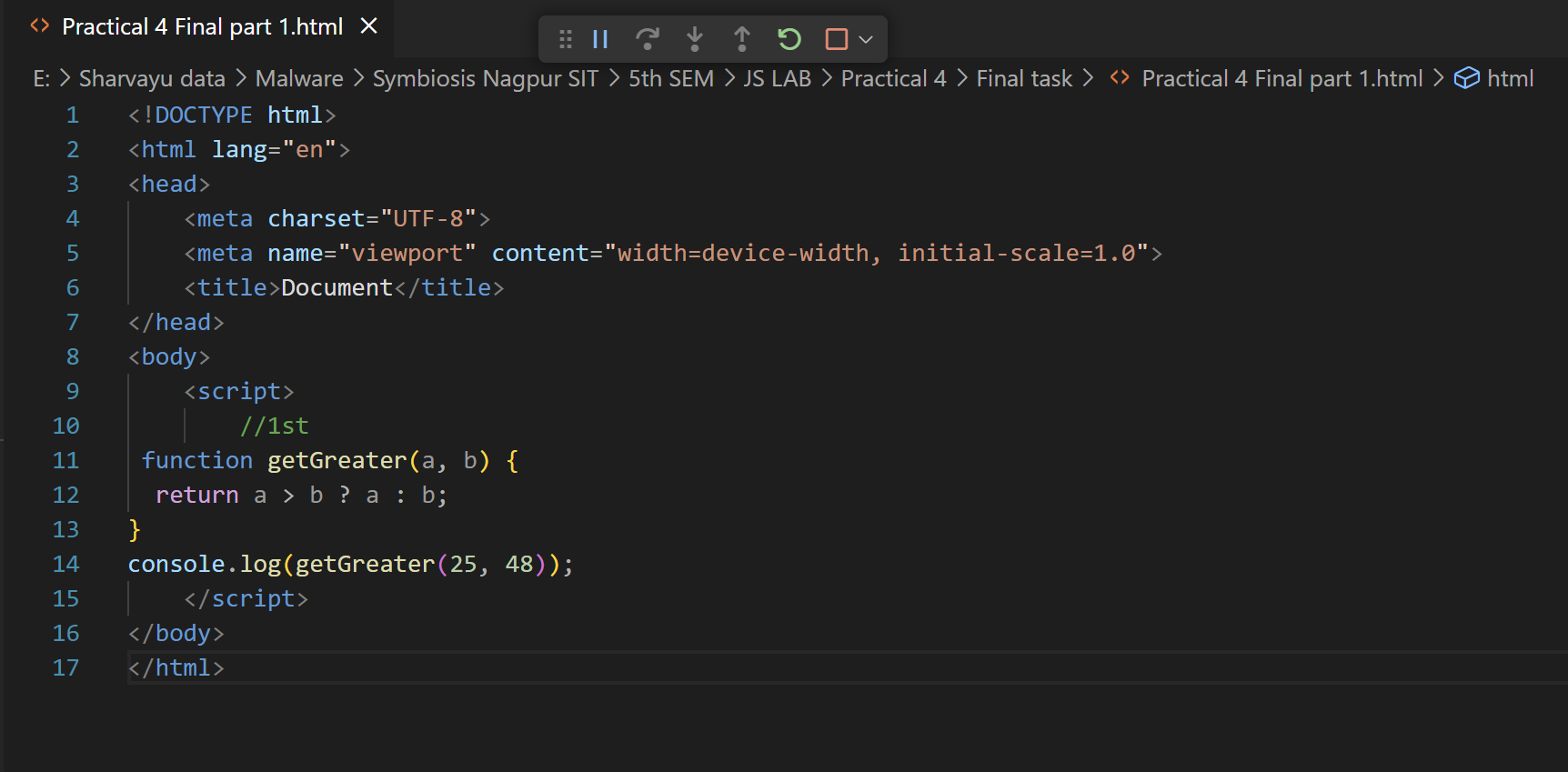
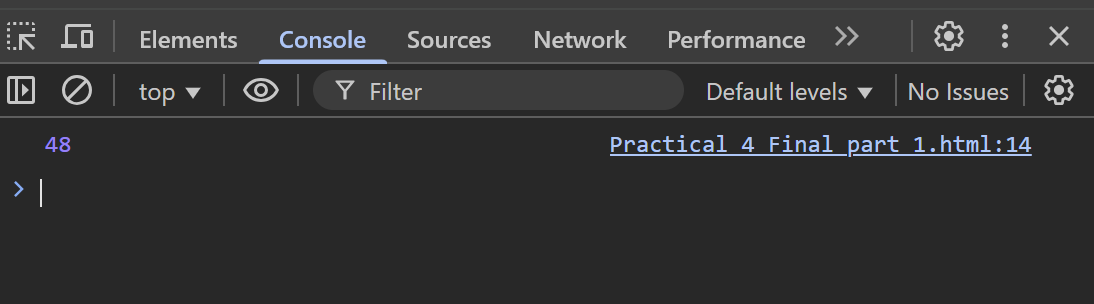
**Part 1 – Functions, Scope, and Closures**

1. Function Declarations

-Write a function getGreater(a, b) that takes two numbers and returns the greater number.

-This will be used to compare two entered PIN attempts and select the stronger one.

Input: getGreater(25, 48)

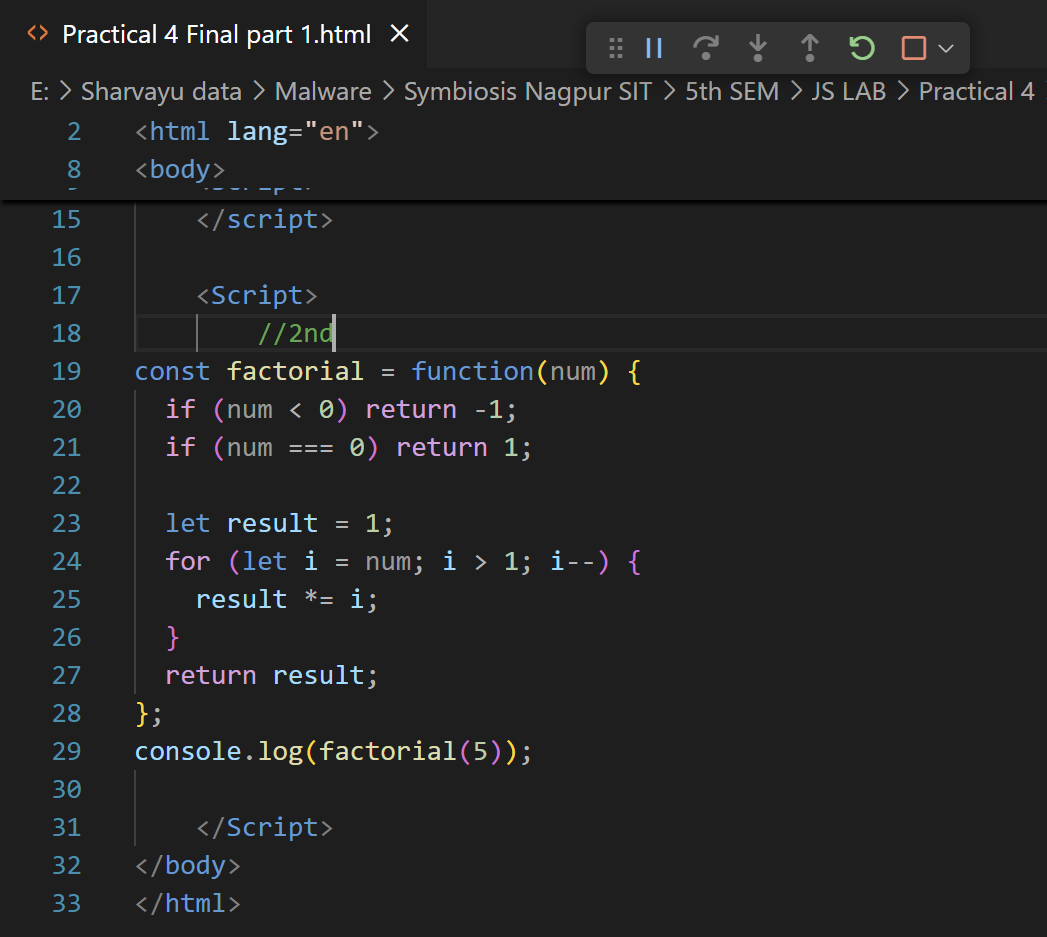
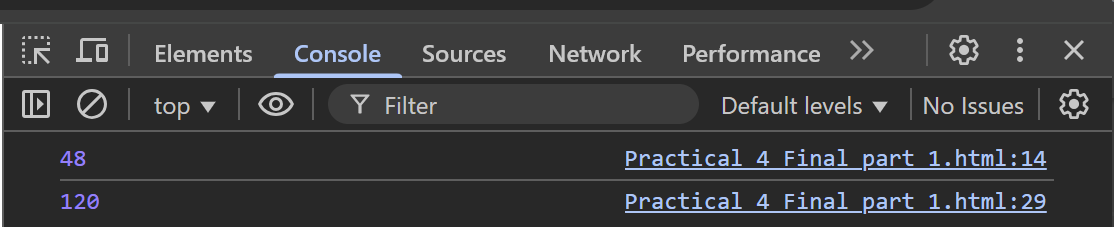
Output: 48 

2. Function Expressions

-Write a function expression factorial that calculates the factorial of a given number.

-The factorial will be used to generate a unique hash code for secret storage.

Input: factorial(5)

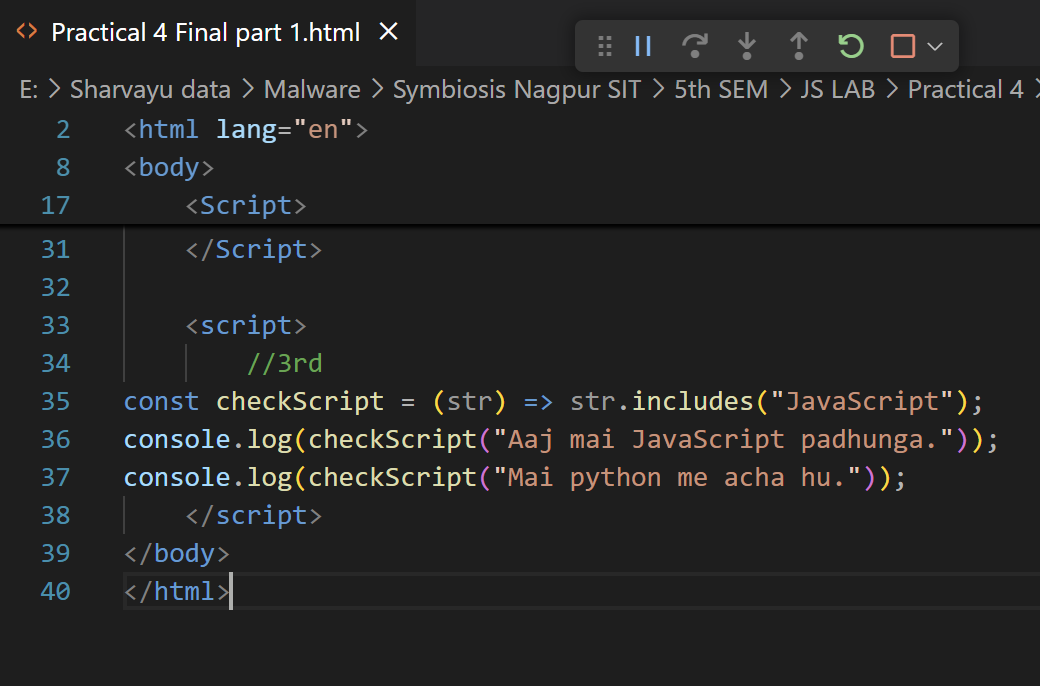
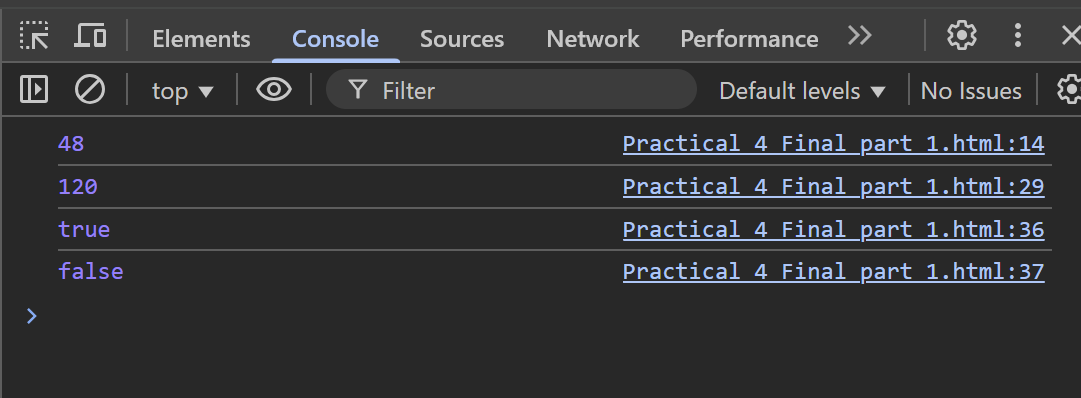
Output: 120 

3. Arrow Functions

-Create an arrow function checkScript(str) that checks if a given string contains the word "JavaScript".

-The Digital Locker will reject notes if they don’t mention JavaScript at least once.

Input: checkScript("I am learning JavaScript today!")

Output: true 

4. Closures

-Create a closure secureMessage(password) that stores a secret message.

-The inner function should return the message only if the given password matches; otherwise, return "Access Denied".

Input:

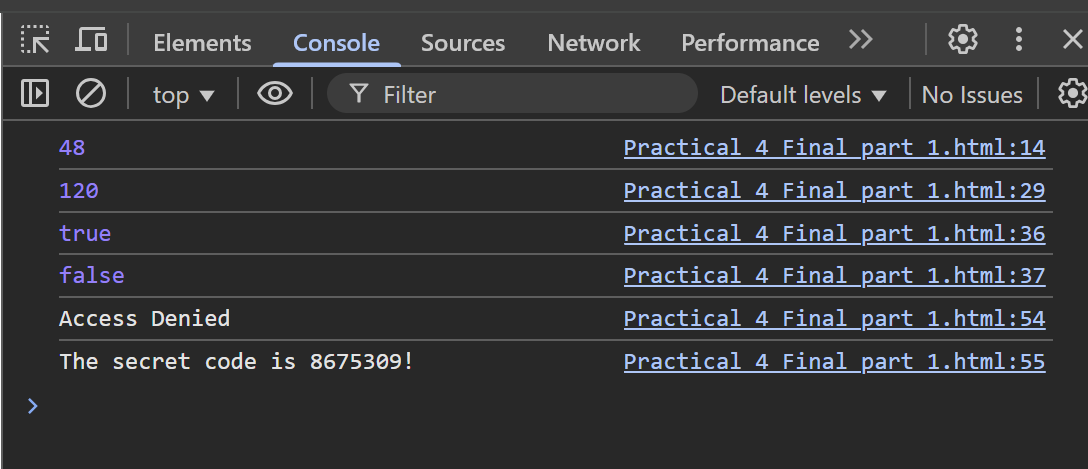
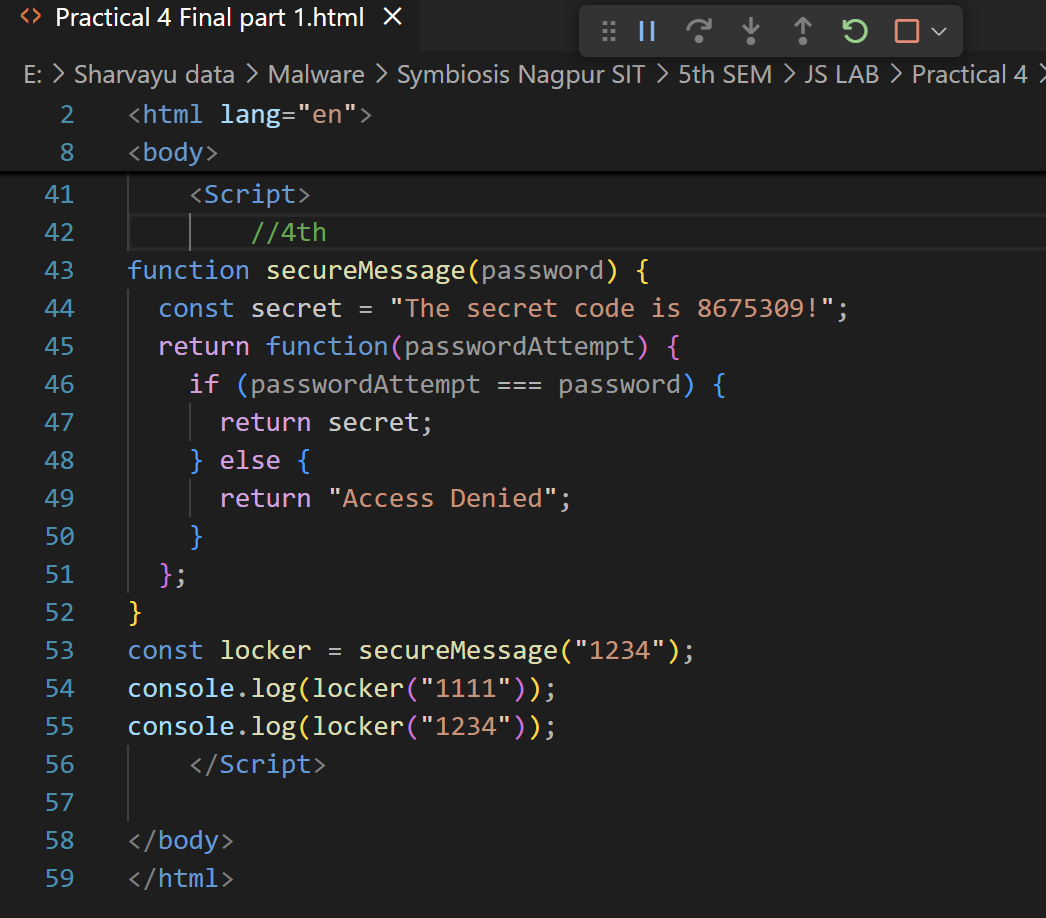
const locker = secureMessage("1234");

console.log(locker("1111"));

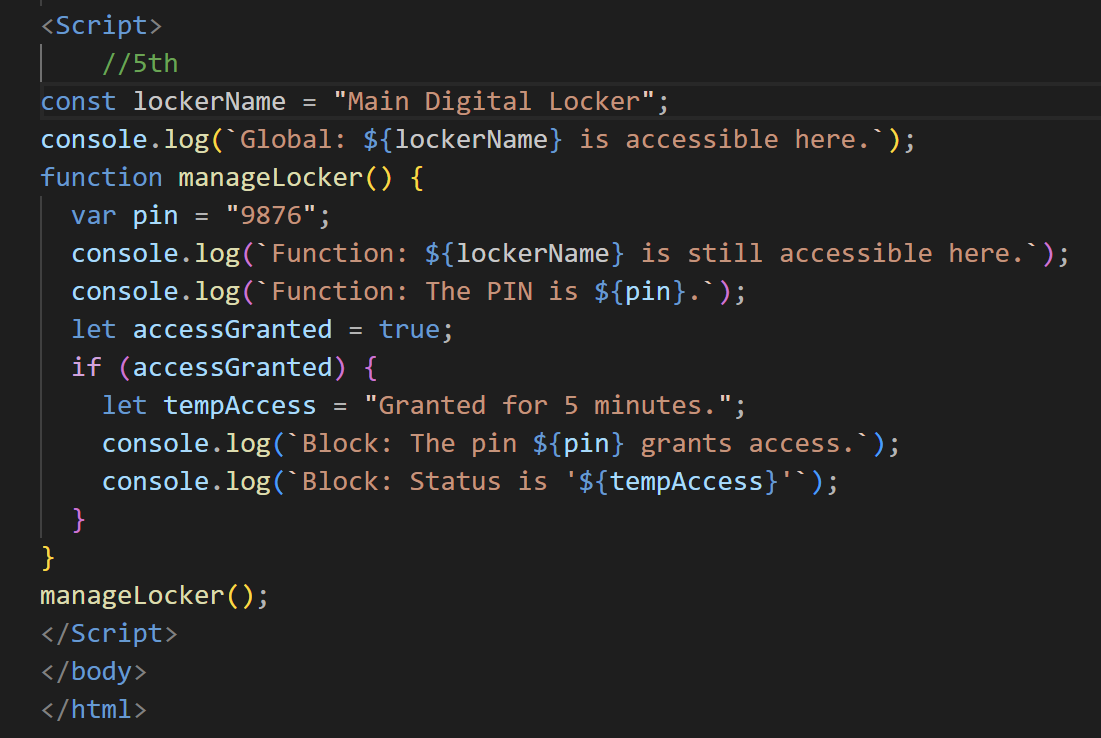
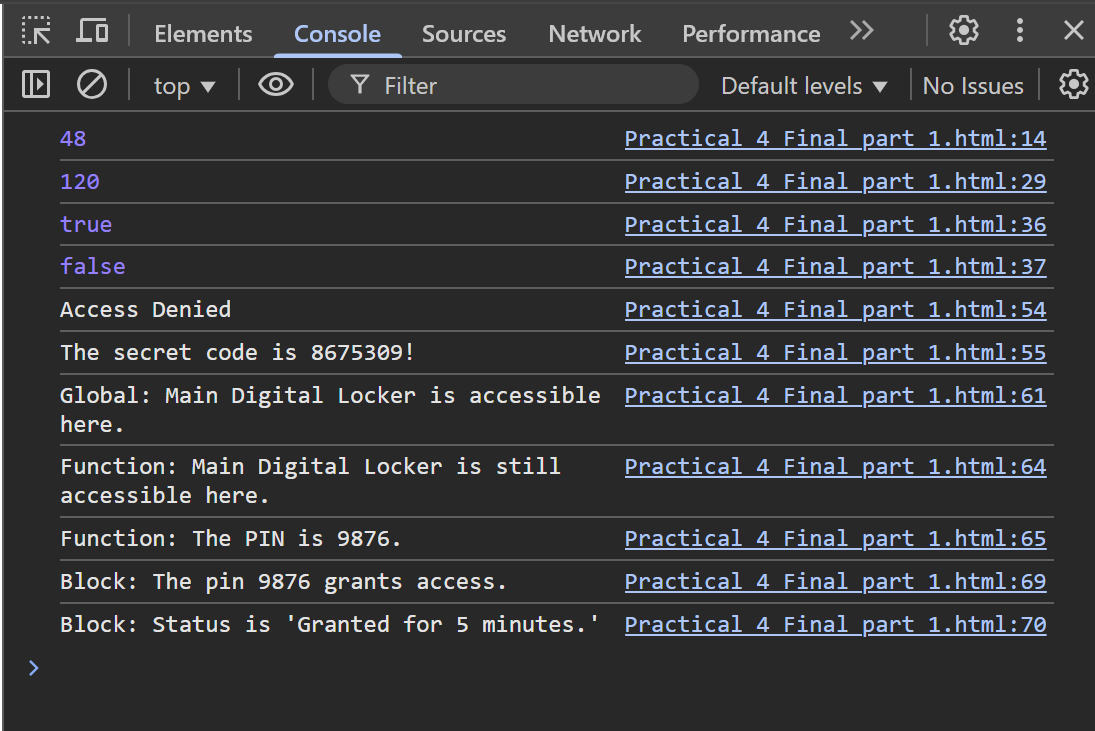
console.log(locker("1234"));

Output:

Access Denied

(Your secret message here)  

5. Scope Check

Demonstrate global scope, function scope, and block scope in your program using meaningful variables like lockerName, pin, and tempAccess. 

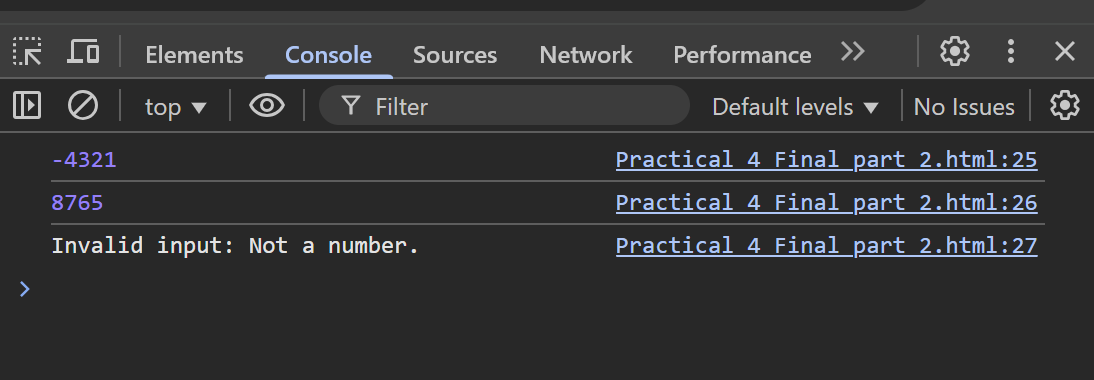
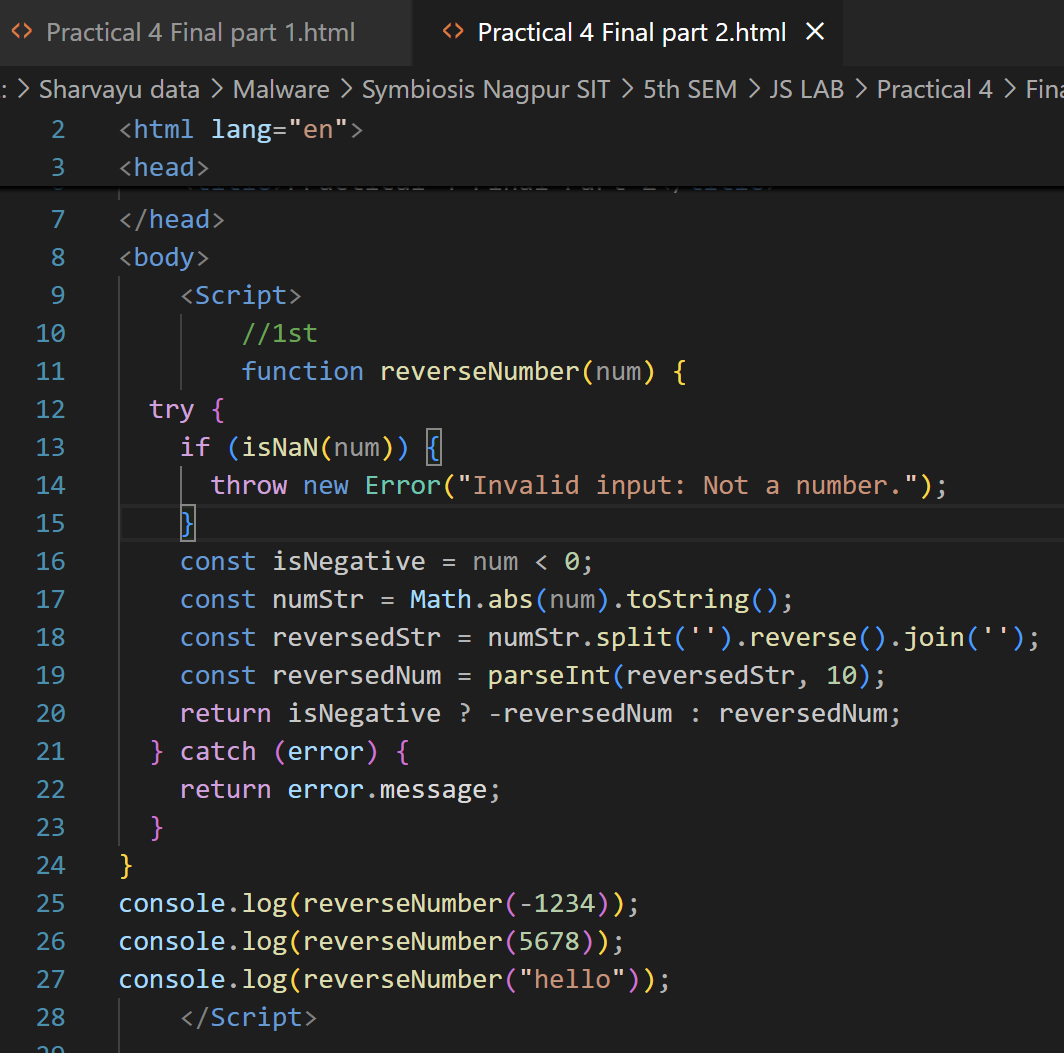
**Part 2 – Error Handling & Palindrome Utilities**

1. Reverse Number with Error Handling

-Write a function reverseNumber(num) that works for both positive and negative numbers.

-Use try...catch to throw an error if input is invalid.

Input: reverseNumber(-1234)

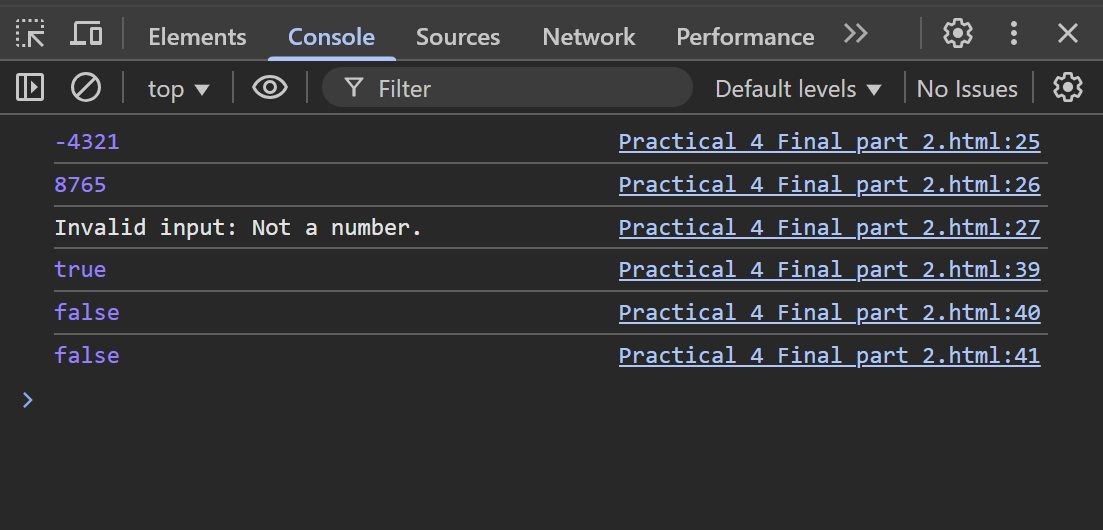
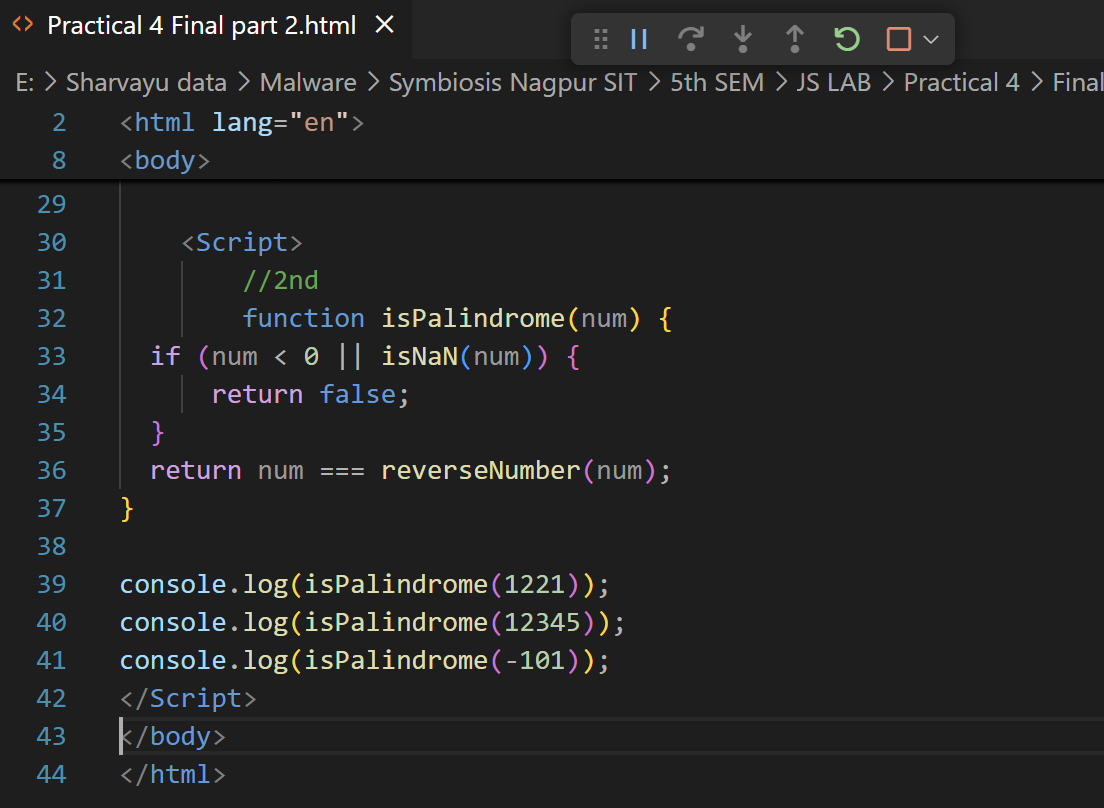
Output: -4321 

2. Palindrome PIN Check

-Write a function isPalindrome(num) that checks whether a number is a palindrome.

-Use it to validate if a user’s PIN is secure (palindromes are considered secure).

Input: isPalindrome(1221)

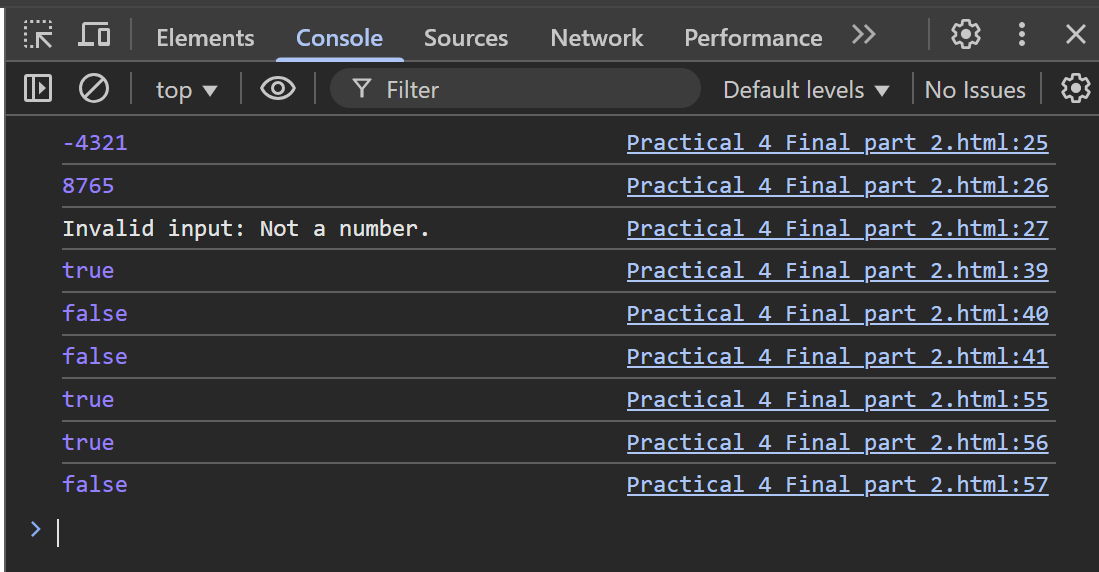
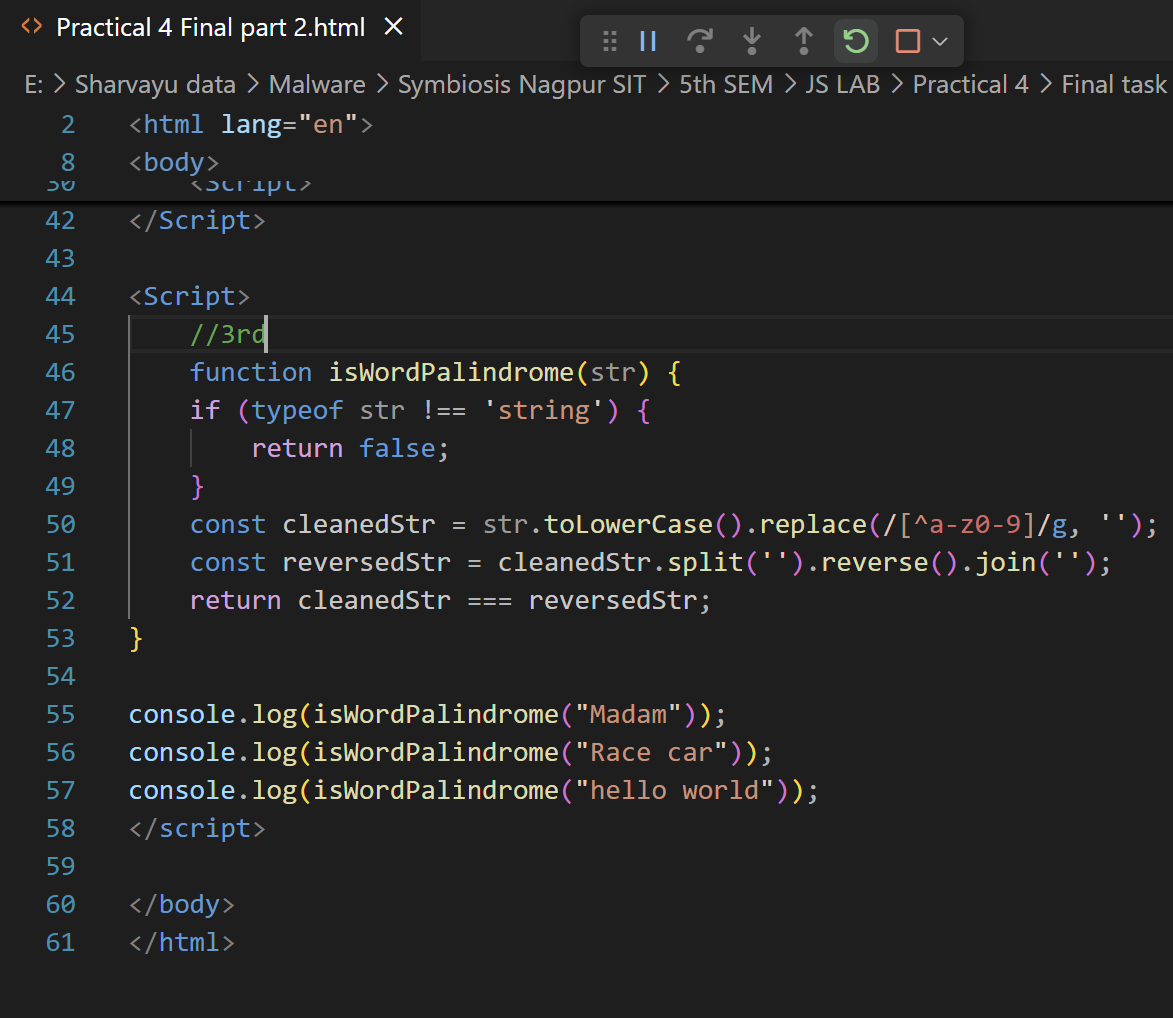
Output: true 

3. Word Palindrome Check

-Extend functionality: accept a string input from the user and check if it is a palindrome (word-based).

-Ignore case and spaces (e.g., "Race car" → palindrome).

Input: "Madam"

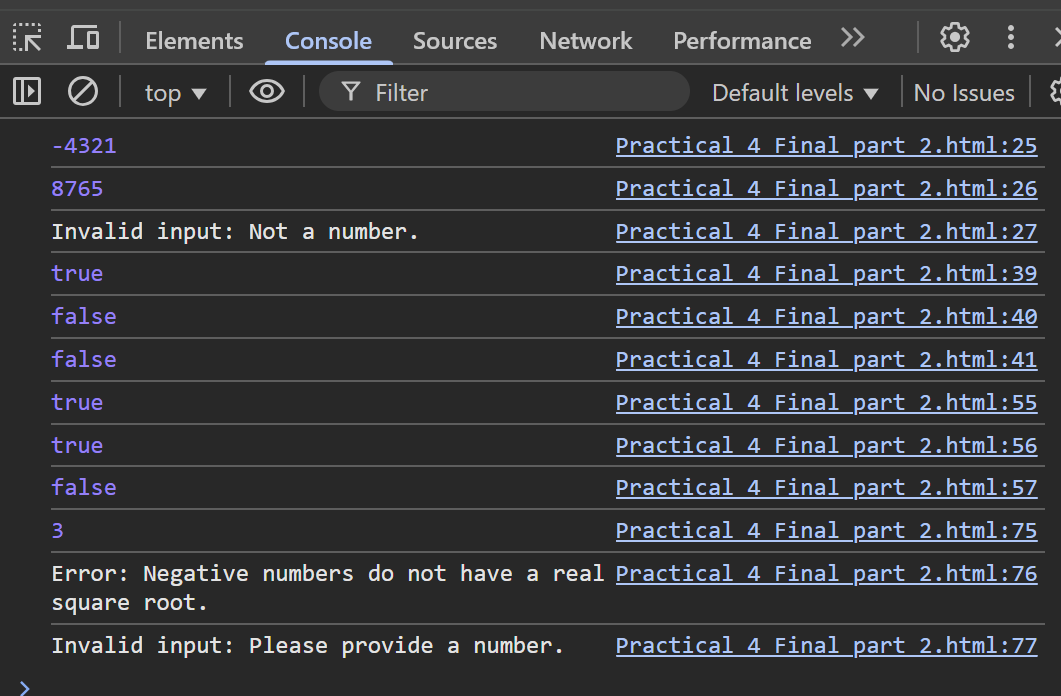
Output: true 

4. Square Root with Error Handling

-Write a function that takes a number and returns its square root. -If input is invalid or negative, throw an error and handle it with try...catch.

Input: sqrt(-9)

Output: "Error: Negative numbers not allowed"

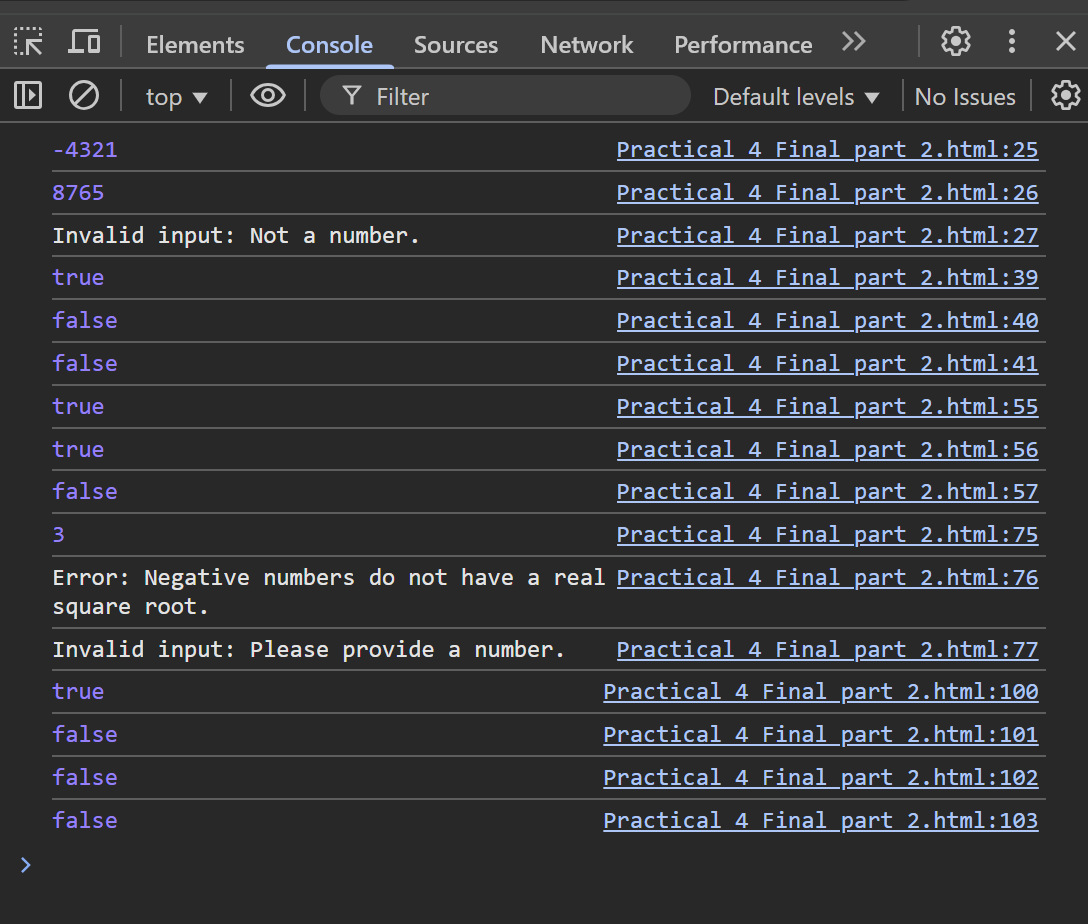
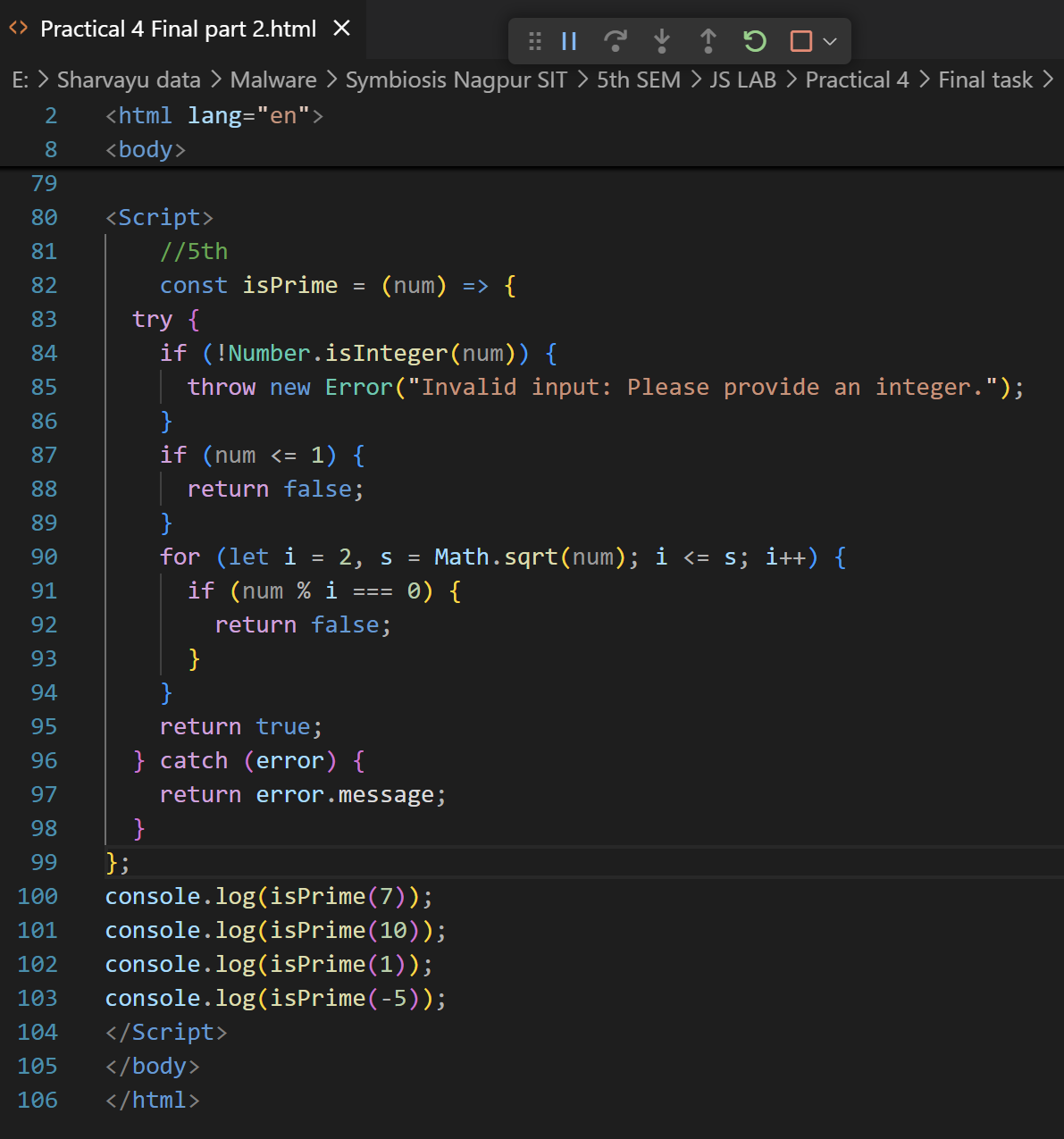


5. Prime Number Checker with Arrow Function

-Create an arrow function isPrime(num) that checks whether a number is prime.

-Use try...catch to handle invalid inputs (e.g., strings or negative numbers).

Input: isPrime(7)

Output: true 

**Final Integration Challenge**

Put it all together in a menu-driven program simulation (console-based). The program should ask the user to choose an option:

1. Compare two PINs and show the stronger one.

2. Generate a factorial hash for a number.

3. Check if a message contains “JavaScript”.

4. Store and access a secret message (closure).

5. Reverse a number and check if it’s a palindrome.

6. Check if a word is a palindrome.

7. Find square root with error handling.

8. Check if a number is prime.

For each option, display proper input prompts, error handling, and results.

Do NOT copy code directly from examples above.

You must adapt logic, rename variables, and integrate into the scenario. If your code looks copy-pasted, it will not be accepted.

