Practical: 1

```
// Create a program that declares and initializes variables of different data
// types (string, number, boolean) and displays their values. Write a function
// that takes two numbers as parameters and returns their sum.

// Declare and initialize variables of different data types
const myString = "Harsh Rana";
const myNumber = 114;
const myBoolean = true;

// Function to calculate the sum of two numbers
function addNumbers(num1, num2) {
  return num1 + num2;
}

// Display values of the variables
console.log("myString:", myString);
console.log("myNumber:", myNumber);
console.log("myBoolean:", myBoolean);

// Testing the function
const num1 = 10;
const num2 = 20;
const sum = addNumbers(num1, num2);
console.log(`The sum of ${num1} and ${num2} is:`, sum);
```

Practical: 2

```
// Create an array of numbers and perform the following operations:
// => Find the length of the array.
// => Access and display specific elements using indexing.
// =>Use array methods like push() ,pop(), shift(), unshift(), join(),
//  delete(),concate(),flat(),splice() and slice() to modify the array.
// Create an object representing a person with properties like name,
// age, and gender. Implement a function that displays the person's details.

// Array operations
const numbersArray = [1, 2, 3, 4, 5];

// Find the length of the array
const arrayLength = numbersArray.length;
console.log("Array length:", arrayLength);

// Access and display specific elements using indexing
console.log("Element at index 0:", numbersArray[0]);
console.log("Element at index 2:", numbersArray[2]);
```

```
// Array methods
numbersArray.push(6); // Add element at the end of the array
console.log("After push:", numbersArray);

numbersArray.pop(); // Remove the last element
console.log("After pop:", numbersArray);

numbersArray.shift(); // Remove the first element
console.log("After shift:", numbersArray);

numbersArray.unshift(0); // Add element at the beginning of the
array
console.log("After unshift:", numbersArray);

const arrayAsString = numbersArray.join(", "); // Convert the
array to a string
console.log("Array as string:", arrayAsString);

// 'delete' is not recommended for arrays as it leaves an
undefined hole, but here's how you would use it:
delete numbersArray[2];
console.log("After delete:", numbersArray);

const secondArray = [7, 8, 9];
const combinedArray = numbersArray.concat(secondArray); //
Concatenate two arrays
console.log("Concatenated array:", combinedArray);

const flattenedArray = combinedArray.flat(); // Flatten nested
arrays
console.log("Flattened array:", flattenedArray);

const splicedArray = flattenedArray.splice(1, 3); // Remove
elements from the array
console.log("After splice:", splicedArray);

const slicedArray = flattenedArray.slice(1, 4); // Extract
elements from the array
console.log("After slice:", slicedArray);

// Object representing a person
const person = {
  name: "Harsh Rana",
  age: 19,
  gender: "Male",
};

// Function to display person's details
function displayPersonDetails(personObj) {
  console.log("Name:", personObj.name);
  console.log("Age:", personObj.age);
```

```
  console.log("Gender:", personObj.gender);
}

// Display person's details
displayPersonDetails(person);
```

Practical: 3

```
// Implement following features of ECMASCRIPT 6.
// • The let keyword
// • The const keyword
// • Arrow Functions
// • The (Spread Of) ... Operator
// • For/of
// • Map Objects
// • Set Objects
// • Classes
// • Promises
// • Symbol
// • Default Parameters
// • Function Rest Parameter

let x = 10;
if (true) {
  let x = 20;
  console.log(x); // Output: 20
}
console.log(x); // Output: 10

const PI = 3.14159;
// PI = 3.14; // This will throw an error, as PI is a constant
and cannot be reassigned.

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

  // Arrow function
  const addArrow = (a, b) => a + b;

console.log(add(2, 3));
console.log(addArrow(2, 3));

const arr1 = [1, 2, 3];
const arr2 = [...arr1, 4, 5];

console.log(arr2);

const arr = [1, 2, 3];
for (const element of arr) {
  console.log(element);
}
```

```
// Output: 1, 2, 3

const myMap = new Map();
myMap.set("name", "Harsh");
myMap.set("age", 19);

console.log(myMap.get("name"));
console.log(myMap.get("age"));

const mySet = new Set();
mySet.add(1);
mySet.add(2);
mySet.add(2); // Ignored, as 2 is already present

console.log(mySet);

const mySet2 = new Set();
mySet2.add(1);
mySet2.add(2);
mySet2.add(2); // Ignored, as 2 is already present

console.log(mySet2);


class Person {
    constructor(name, age) {
      this.name = name;
      this.age = age;
    }

    sayHello() {
      console.log(`Hello, my name is ${this.name} and I am
${this.age} years old.`);
    }
  }

  const Harsh = new Person("Harsh", 19);
  Harsh.sayHello();


  const fetchData = () => {
    return new Promise((resolve, reject) => {
      setTimeout(() => {
        resolve("Data fetched successfully!");
      }, 2000);
    });
  };

  fetchData()
    .then((data) => console.log(data))
    .catch((error) => console.error(error));
```

```
const mySymbol = Symbol("mySymbol");
const obj = {
  [mySymbol]: "This is a symbol key",
};

console.log(obj[mySymbol]);


function greet(name = "Guest") {
    console.log(`Hello, ${name}!`);
  }

  greet();
  greet("Harsh");


  function sum(...numbers) {
    return numbers.reduce((acc, num) => acc + num, 0);
  }

  console.log(sum(1, 2, 3, 4, 5));


  function sum(...numbers) {
    return numbers.reduce((acc, num) => acc + num, 0);
  }

  console.log(sum(1, 2, 3, 4, 5));
```

Practical: 4

```
Write a function that calculates the factorial of a given number
using recursion.
Create a nested function that performs a specific task and
invoke it within another function.
(NOTE: Implement the concept of variable scope in functions by
declaring
  variables with different scopes (global, local) and accessing
them).

let globalVariable = "I am a global variable";

function factorialRecursive(number) {

  let localVariable = "I am a local variable";

  if (number === 0 || number === 1) {
    return 1;
  } else {
    return number * factorialRecursive(number - 1);
  }
}
```

```
// Function to demonstrate accessing variables with different
scopes
function variableScopeDemo() {
  console.log("Accessing the global variable:", globalVariable);
  console.log("Trying to access the local variable:",
localVariable); //Error: localVariable is not defined in this
scope
}

const num = 5;
console.log(`Factorial of ${num} is:`, factorialRecursive(num));

variableScopeDemo();
```