Array and Object - Assignment

1. In the following shopping cart add, remove, and edit items

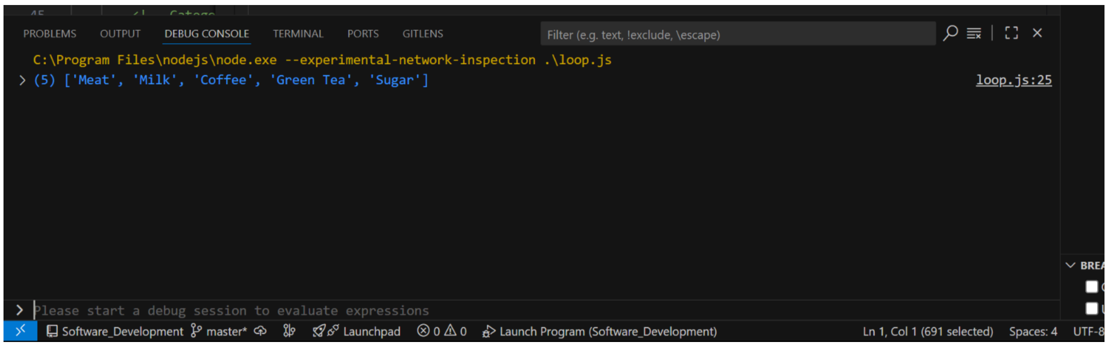
=> const shoppingCart = ['Milk', 'Coffee', 'Tea', 'Honey']

- add 'Meat' in the beginning of your shopping cart if it has not been already
- add Sugar at the end of you shopping cart if it has not been already
- remove 'Honey' if you are allergic to honey
- modify Tea to 'Green Tea'

PROGRAM:

```
// Initializing shopping cart array
let shoppingCart = ['Milk', 'Coffee', 'Tea', 'Honey'];

// Adding 'Meat' to the beginning if not already present
if(!shoppingCart.includes('Meat')){
    shoppingCart.unshift('Meat');
}

// Adding 'Sugar' to the end if not already present
if(!shoppingCart.includes('Sugar')){
    shoppingCart.push('Sugar');
}

// Removing 'Honey' if allergic
const index = shoppingCart.indexOf('Honey');
if(index > -1){
    shoppingCart.splice(index, 1);
}
// Modifying 'Tea' to 'Green Tea'
const teaIndex = shoppingCart.indexOf('Tea');
if(teaIndex > -1){
    shoppingCart[teaIndex]= 'Green Tea';
}
// final shopping cart
console.log(shoppingCart);
```

RESULT:



```
C:\Program Files\nodejs\node.exe --experimental-network-inspection .\loop.js
> (5) ['Meat', 'Milk', 'Coffee', 'Green Tea', 'Sugar']                loop.js:25
```

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
Filter (e.g. text, !exclude, \escape)
Please start a debug session to evaluate expressions
Software_Development
master*
Launchpad
Launch Program (Software_Development)
Ln 1, Col 1 (691 selected)
Spaces: 4
UTF-8

---

2. The following is an array of 10 students ages:

   `const ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]`

Instructions:

- Sort the array and find the min and max age.
- Find the median age.
- Find the average age.
- Find the range of the ages (max minus min).
- Compare the value of (min - average) and (max - average), using the absolute value (`abs()`) method.

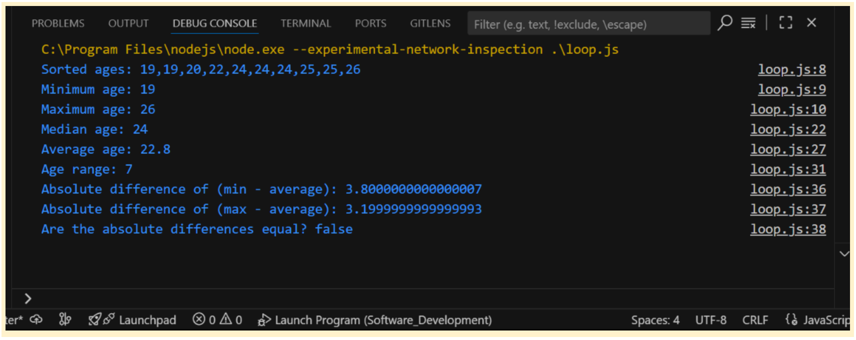
PROGRAM:

```
// 2. The following is an array of 10 students ages:

const ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24];



// Sort the array and find the min and max age.

ages.sort((a, b) => a - b);

const minAge = ages[0];

const maxAge = ages[ages.length - 1];
```

```
console.log(`Sorted ages: ${ages}`);

console.log(`Minimum age: ${minAge}`);

console.log(`Maximum age: ${maxAge}`);



// Find the median age

let medianAge;

const mid = Math.floor(ages.length / 2);

if (ages.length % 2 === 0) {

    // Even number of elements, take the average of the two
middle items

    medianAge = (ages[mid - 1] + ages[mid]) / 2;

} else {

    // Odd number of elements, take the middle item

    medianAge = ages[mid];

}

console.log(`Median age: ${medianAge}`);



// Find the average age

const sumOfAges = ages.reduce((sum, age) => sum + age, 0);

const averageAge = sumOfAges / ages.length;

console.log(`Average age: ${averageAge}`);



// Find the range of the ages

const ageRange = maxAge - minAge;

console.log(`Age range: ${ageRange}`);
```

```javascript
// Compare the value of (min - average) and (max - average)
const minAvgDiff = Math.abs(minAge - averageAge);
const maxAvgDiff = Math.abs(maxAge - averageAge);
console.log(`Absolute difference of (min - average): ${minAvgDiff}`);
console.log(`Absolute difference of (max - average): ${maxAvgDiff}`);
console.log(`Are the absolute differences equal? ${minAvgDiff === maxAvgDiff}`);
```

RESULT:



```
C:\Program Files\nodejs\node.exe --experimental-network-inspection .\loop.js
Sorted ages: 19,19,20,22,24,24,24,25,25,26                          loop.js:8
Minimum age: 19                                                      loop.js:9
Maximum age: 26                                                     loop.js:10
Median age: 24                                                      loop.js:22
Average age: 22.8                                                   loop.js:27
Age range: 7                                                        loop.js:31
Absolute difference of (min - average): 3.8000000000000007          loop.js:36
Absolute difference of (max - average): 3.1999999999999993          loop.js:37
Are the absolute differences equal? false                           loop.js:38
```

ter* Launchpad ⊗ 0 ⚠ 0 Launch Program (Software_Development) Spaces: 4 UTF-8 CRLF JavaScrip

---

3. Object Extensibility and Sealing

   a) Use the Object.preventExtensions method to prevent any further additions of properties to the student object.

   b) Use the Object.isExtensible method to check if the student object is extensible. Store the result in a variable called extensibleStatus.

   c) Create a new object called teacher with a 'subject' property set to 'Math'.

   d) Use the Object.seal method to seal the teacher object, preventing any additions or deletions of properties.

e) Use the Object.isSealed method to check if the teacher object is sealed. Store the result in a variable called sealedStatus.

f) Print the extensibleStatus and sealedStatus to the console.

PROGRAM :

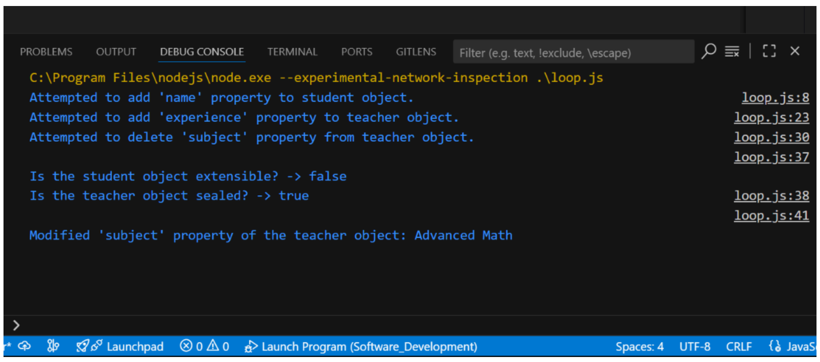
```
const student = {

    age: 20

};

Object.preventExtensions(student);



try {

  student.name = "Alice";

  console.log("Attempted to add 'name' property to student
object.");

} catch (e) {

  console.error("Failed to add 'name' property:", e.message);

}



const extensibleStatus = Object.isExtensible(student);



const teacher = {

    subject: 'Math'

};



Object.seal(teacher);
```

```javascript
try {

  teacher.experience = 5;

  console.log("Attempted to add 'experience' property to teacher
object.");

} catch (e) {

  console.error("Failed to add 'experience' property:",
e.message);

}



try {

  delete teacher.subject;

  console.log("Attempted to delete 'subject' property from
teacher object.");

} catch (e) {

  console.error("Failed to delete 'subject' property:",
e.message);

}



const sealedStatus = Object.isSealed(teacher);



console.log(`\nIs the student object extensible? ->
${extensibleStatus}`);

console.log(`Is the teacher object sealed? -> ${sealedStatus}`);



teacher.subject = "Advanced Math";

console.log(`\nModified 'subject' property of the teacher object:
${teacher.subject}`);
```

Summary of Differences

- `Object.preventExtensions()`: Prevents **additions** of new properties.
- `Object.seal()`: Prevents **additions** and **deletions** of properties, but **allows modifications** of existing ones.

RESULT :


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
Filter (e.g. text, !exclude, \escape)
Launchpad
Launch Program (Software_Development)
Spaces: 4
UTF-8
CRLF

```
C:\Program Files\nodejs\node.exe --experimental-network-inspection .\loop.js
Attempted to add 'name' property to student object.                    loop.js:8
Attempted to add 'experience' property to teacher object.               loop.js:23
Attempted to delete 'subject' property from teacher object.             loop.js:30
                                                                        loop.js:37
Is the student object extensible? -> false
Is the teacher object sealed? -> true                                   loop.js:38
                                                                        loop.js:41
Modified 'subject' property of the teacher object: Advanced Math
```

---

4. Assignment: Building a Student Management System

   Description: You are tasked with building a student management system using JavaScript. The system should allow you to perform various operations on a list of students, including adding, updating, deleting, and displaying student information.

   Implement the following functions using pure JavaScript (without any external libraries or frameworks):

   a. Add a Student: Create a function to add a new student to the array.

   b. Update Student Information: Create a function to update a student's information based on their id.

   c. Delete a Student: Create a function to delete a student based on their id.

   d. List All Students: Create a function to display a list of all students.

   e. Find Students by Grade: Create a function to find all students who have a specific grade.

   f. Calculate Average Age: Create a function to calculate the average age of all students using array method.

PROGRAM :

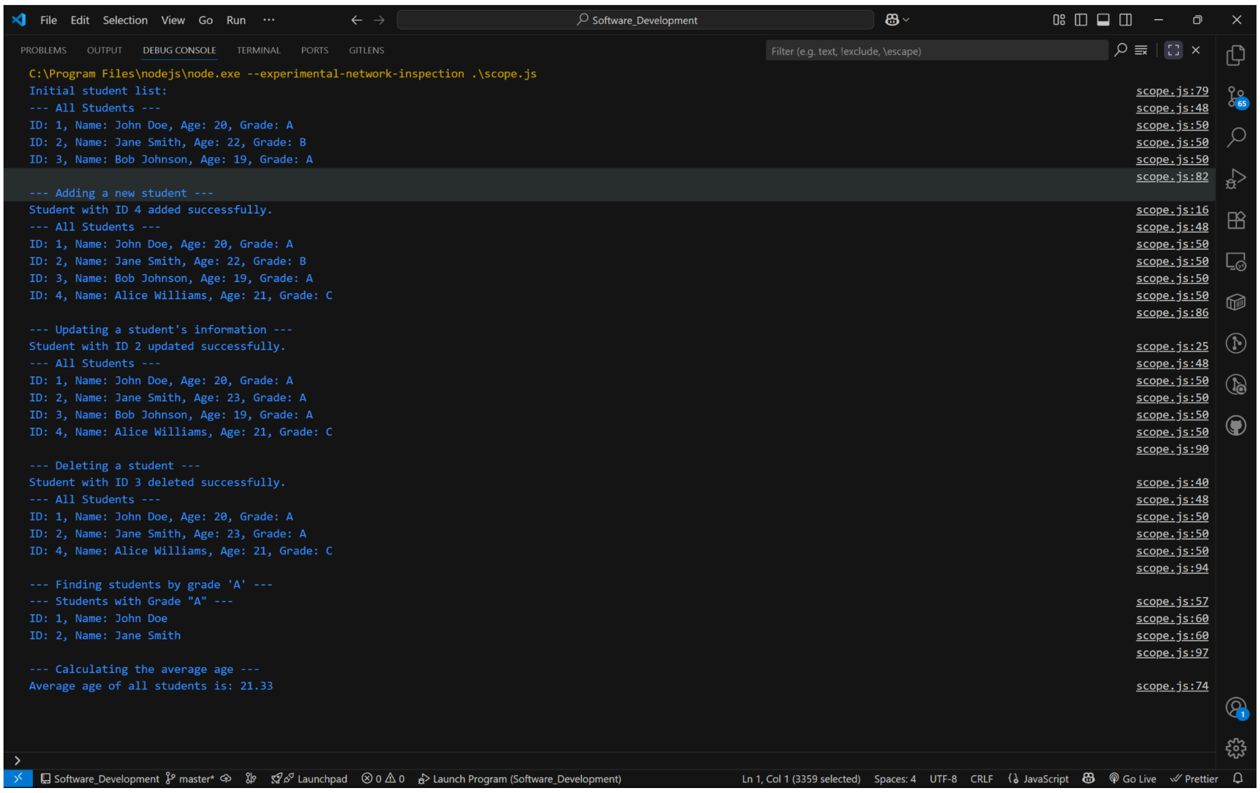
```
const students = [

  { id: 1, firstName: "John", lastName: "Doe", age: 20, grade:
"A" },

  { id: 2, firstName: "Jane", lastName: "Smith", age: 22, grade:
"B" },

  { id: 3, firstName: "Bob", lastName: "Johnson", age: 19, grade:
"A" },

];



// a. Add a Student

function addStudent(student) {

  // Check if a student with the same id already exists

  const existingStudent = students.find(s => s.id ===
student.id);

  if (existingStudent) {

    console.log(`Error: Student with ID ${student.id} already
exists.`);

    return;

  }

  students.push(student);

  console.log(`Student with ID ${student.id} added
successfully.`);

}



// b. Update Student Information
```

```
function updateStudent(id, updatedInfo) {

  const index = students.findIndex(s => s.id === id);

  if (index !== -1) {

    // Merge the existing student object with the updated
information

    students[index] = { ...students[index], ...updatedInfo };

    console.log(`Student with ID ${id} updated successfully.`);

  } else {

    console.log(`Error: Student with ID ${id} not found.`);

  }

}



// c. Delete a Student

function deleteStudent(id) {

  const initialLength = students.length;

  // Filter out the student with the specified id

  const updatedStudents = students.filter(s => s.id !== id);



  if (updatedStudents.length < initialLength) {

    // Overwrite the original array with the new one

    students.splice(0, students.length, ...updatedStudents);

    console.log(`Student with ID ${id} deleted successfully.`);

  } else {

    console.log(`Error: Student with ID ${id} not found.`);

  }
}



// d. List All Students

function listAllStudents() {

  console.log("--- All Students ---");

  students.forEach(student => {

    console.log(`ID: ${student.id}, Name: ${student.firstName}
${student.lastName}, Age: ${student.age}, Grade:
${student.grade}`);

  });

}



// e. Find Students by Grade

function findStudentsByGrade(grade) {

  const matchingStudents = students.filter(student =>
student.grade === grade);

  console.log(`--- Students with Grade "${grade}" ---`);

  if (matchingStudents.length > 0) {

    matchingStudents.forEach(student => {

      console.log(`ID: ${student.id}, Name: ${student.firstName}
${student.lastName}`);

    });

  } else {

    console.log(`No students found with grade "${grade}".`);

  }

}
```

```
// f. Calculate Average Age

function calculateAverageAge() {

  if (students.length === 0) {

    return 0; // Return 0 if there are no students

  }

  const totalAge = students.reduce((sum, student) => sum +
student.age, 0);

  const averageAge = totalAge / students.length;

  console.log(`Average age of all students is:
${averageAge.toFixed(2)}`);

  return averageAge;

}



// Example usage of the functions

console.log("Initial student list:");

listAllStudents();



console.log("\n--- Adding a new student ---");

addStudent({ id: 4, firstName: "Alice", lastName: "Williams",
age: 21, grade: "C" });

listAllStudents();



console.log("\n--- Updating a student's information ---");

updateStudent(2, { age: 23, grade: "A" });

listAllStudents();
```

```
console.log("\n--- Deleting a student ---");

deleteStudent(3);

listAllStudents();



console.log("\n--- Finding students by grade 'A' ---");

findStudentsByGrade("A");



console.log("\n--- Calculating the average age ---");

calculateAverageAge();
```

RESULT :



```
C:\Program Files\nodejs\node.exe --experimental-network-inspection .\scope.js
Initial student list:                                                    scope.js:79
--- All Students ---                                                     scope.js:48
ID: 1, Name: John Doe, Age: 20, Grade: A                                 scope.js:50
ID: 2, Name: Jane Smith, Age: 22, Grade: B                               scope.js:50
ID: 3, Name: Bob Johnson, Age: 19, Grade: A                              scope.js:50
                                                                         scope.js:82
--- Adding a new student ---
Student with ID 4 added successfully.                                    scope.js:16
--- All Students ---                                                     scope.js:48
ID: 1, Name: John Doe, Age: 20, Grade: A                                 scope.js:50
ID: 2, Name: Jane Smith, Age: 22, Grade: B                               scope.js:50
ID: 3, Name: Bob Johnson, Age: 19, Grade: A                              scope.js:50
ID: 4, Name: Alice Williams, Age: 21, Grade: C                           scope.js:50
                                                                         scope.js:86
--- Updating a student's information ---
Student with ID 2 updated successfully.                                  scope.js:25
--- All Students ---                                                     scope.js:48
ID: 1, Name: John Doe, Age: 20, Grade: A                                 scope.js:50
ID: 2, Name: Jane Smith, Age: 23, Grade: A                               scope.js:50
ID: 3, Name: Bob Johnson, Age: 19, Grade: A                              scope.js:50
ID: 4, Name: Alice Williams, Age: 21, Grade: C                           scope.js:50
                                                                         scope.js:90
--- Deleting a student ---
Student with ID 3 deleted successfully.                                  scope.js:40
--- All Students ---                                                     scope.js:48
ID: 1, Name: John Doe, Age: 20, Grade: A                                 scope.js:50
ID: 2, Name: Jane Smith, Age: 23, Grade: A                               scope.js:50
ID: 4, Name: Alice Williams, Age: 21, Grade: C                           scope.js:50
                                                                         scope.js:94
--- Finding students by grade 'A' ---
--- Students with Grade "A" ---                                          scope.js:57
ID: 1, Name: John Doe                                                    scope.js:60
ID: 2, Name: Jane Smith                                                  scope.js:60
                                                                         scope.js:97
--- Calculating the average age ---
Average age of all students is: 21.33                                    scope.js:74
```

---

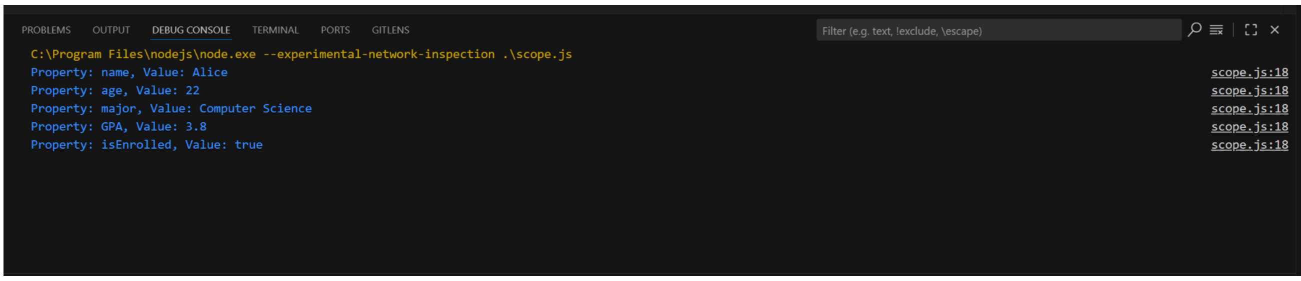
5. You are given a JavaScript object representing a student's information. Your task is to use the 'for...in' loop to iterate over the properties of the object and perform various operations.

a) Create a function displayStudentInfo that takes the student object as a parameter. Inside this function, use a 'for...in' loop to iterate over the properties of the student object and print each property and its corresponding value to the console.

PROGRAM :

```
const student = {

  name: "Alice",

  age: 22,

  major: "Computer Science",

  GPA: 3.8,

  isEnrolled: true,

};

function displayStudentInfo(studentObject) {

  // Use a for...in loop to iterate over the properties of the object

  for (const property in studentObject) {

    // Check if the property is an own property of the object

    if (Object.prototype.hasOwnProperty.call(studentObject, property)) {

      console.log(`Property: ${property}, Value: ${studentObject[property]}`);

    }

  }

}

// Call the function to display the student's information

displayStudentInfo(student);
```

RESULT :



```
C:\Program Files\nodejs\node.exe --experimental-network-inspection .\scope.js
Property: name, Value: Alice                                          scope.js:18
Property: age, Value: 22                                              scope.js:18
Property: major, Value: Computer Science                              scope.js:18
Property: GPA, Value: 3.8                                             scope.js:18
Property: isEnrolled, Value: true                                     scope.js:18
```