**Question 1**

Implement a data structure for a product catalog using JavaScript and create queries to retrieve data as per following specifications:  
1. Create an array to hold an inventory of store items.

2. Create three items, each having the properties of name, model, cost, and quantity.

3. Add all three objects to the main array using an array method, and then log the inventory array to the console.

4. Access the quantity element of your third item, and log it to the console. Experiment by adding and accessing more elements within your data structure.

**Source Code**

let inventory = [];

const item1 = {

name: "Laptop",

model: "XPro-15",

cost: 1299.99,

quantity: 25

};

const item2 = {

name: "Smartphone",

model: "Galaxy-Z",

cost: 799.99,

quantity: 50

};

const item3 = {

name: "Headphones",

model: "AudioMax-300",

cost: 149.99,

quantity: 100

};

inventory.push(item1, item2, item3);

console.log("Complete Inventory:", inventory);

console.log("Quantity of third item (Headphones):", inventory[2].quantity);

const item4 = {

name: "Tablet",

model: "Tab-S8",

cost: 499.99,

quantity: 35

};

inventory.push(item4);

function getItemByName(itemName) {

return inventory.find(item => item.name === itemName);

}

function getTotalInventoryValue() {

return inventory.reduce((total, item) => total + (item.cost \* item.quantity), 0);

}

function getItemsBelowQuantity(threshold) {

return inventory.filter(item => item.quantity < threshold);

}

console.log("\nAdditional Queries:");

console.log("Laptop details:", getItemByName("Laptop"));

console.log("Total inventory value: $", getTotalInventoryValue().toFixed(2));

console.log("Items with quantity below 40:", getItemsBelowQuantity(40));

console.log("\nAccessing specific elements:");

console.log("Second item's model:", inventory[1].model);

console.log("Fourth item's cost:", inventory[3].cost);

console.log("First item's quantity:", inventory[0].quantity);

**Output**

*Terminal*

**Question 2**

Implement using JavaScript a Rock-Paper-Scissors game between a player and the computer, where both will make a random selection of either Rock, Paper, or Scissors. Rock will beat out Scissors, Paper will beat out Rock, and Scissors will beat out Paper. Use the following steps for your implementation of the game.

1. Create an array that contains the variables Rock, Paper, and Scissors.

2. Set up a variable that generates a random number 0-2 for the player and then do the same for the computer's selection. The number represents the index values in the array of the 3 items.

3. Create a variable to hold a response message to the user. This can show the random results for the player and then also the result for the computer of the matching item from the array.

4. Create a condition to handle the player and computer selections. If both are the same, this results in a tie.

5. Use conditions to apply the game logic and return the correct results. There are several ways to do this with the condition statements, but you could check which player's index value is bigger and assign the victory accordingly, with the exception of Rock beating Scissors.

6. Add a new output message that shows the player selection versus the computer selection and the result of the game.

**Source Code**

const readline = require('readline').createInterface({

input: process.stdin,

output: process.stdout

});

const choices = ["Rock", "Paper", "Scissors"];

readline.question("Enter your choice (Rock, Paper, or Scissors): ", (userInput) => {

const playerIndex = choices.findIndex(choice => choice.toLowerCase() === userInput.trim().toLowerCase());

const computerIndex = Math.floor(Math.random() \* 3);

if (playerIndex === -1) {

console.log("Invalid choice! Please enter Rock, Paper, or Scissors.");

} else {

let responseMessage = `Player chose: ${choices[playerIndex]}\nComputer chose: ${choices[computerIndex]}`;

let result;

if (playerIndex === computerIndex) {

result = "It's a tie!";

} else if (

(playerIndex === 0 && computerIndex === 2) || // Rock beats Scissors

(playerIndex === 1 && computerIndex === 0) || // Paper beats Rock

(playerIndex === 2 && computerIndex === 1) // Scissors beats Paper

) {

result = "Player wins!";

} else {

result = "Computer wins!";

}

const finalMessage = `${responseMessage}\nResult: ${result}`;

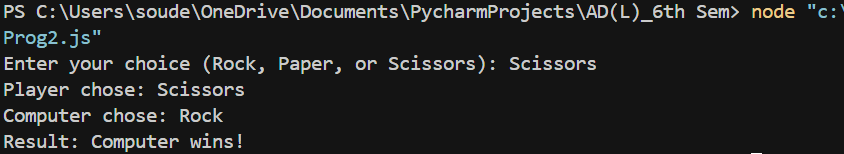
console.log(finalMessage);

}

readline.close();

});

**Output**

*Terminal*

**Question 3**

Implement using JavaScript arrow notation for functions ( ( ) => { ... } ) two functions that output the values One and Two to the console, respectively. Create a third function that outputs the value Three to the console, and then invokes the first two functions. Create a fourth function that outputs the word Four to the console and also use setTimeout() to invoke the first function immediately and then the third function. What does your output look like in the console? Try to get the console to output the following:  
Four  
Three  
One  
Two  
One

**Source Code**

const outputOne = () => console.log("One");

const outputTwo = () => console.log("Two");

const outputThree = () => {

console.log("Three");

outputOne();

outputTwo();

};

const outputFour = () => {

console.log("Four");

setTimeout(() => {

outputThree();

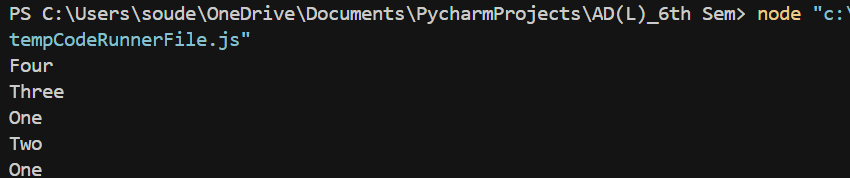
setTimeout(outputOne, 0);

}, 0);

};

outputFour();

**Output**

*Terminal*

**Question 4**

Implement using JavaScript a class to track the employees of a company as per the following specifications:  
1. Use first names, last names, and the number of years worked as values in the constructor.

2. Create two or more people with values for their first names, last names, and the number of years they've worked at the company. Add the people into an array.

3. Set up a prototype to return the details of the person's first and last names and how long they've worked at the company.

4. Iterate the contents of the array to output the results into the console, adding some text to make the output a full sentence

**Source Code**

class Employee {

constructor(firstName, lastName, yearsWorked) {

this.firstName = firstName;

this.lastName = lastName;

this.yearsWorked = yearsWorked;

}

getDetails() {

return `${this.firstName} ${this.lastName} has worked at the company for ${this.yearsWorked} years`;

}

}

const employees = [

new Employee("John", "Smith", 5),

new Employee("Maria", "Gonzalez", 3),

new Employee("Alex", "Johnson", 8)

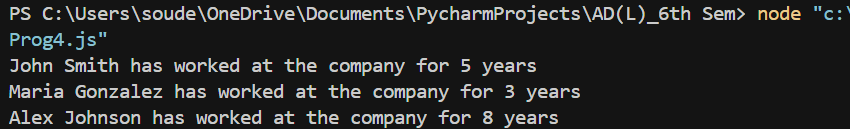
];

employees.forEach(employee => {

console.log(employee.getDetails());

});

**Output**

*Terminal*

**Question 5**

Implement using JavaScript a class which will allow you to work out the combined price of a number of items, and interact with it to work out the total cost of different orders.

1. Create a class that contains the prices of two menu items as private field declarations.

2. Use the constructor in the class to get the argument values (how many of each item are being bought).

3. Create a method to calculate and return the total cost depending on how many of each item the user selects.

4. Use a getter property to grab the value output by the calculation method.

5. Create two or three objects with different combinations of menu selections, and output the total cost in the console.

**Source Code**

class Order {

#burgerPrice = 5.99;

#friesPrice = 2.49;

constructor(numBurgers, numFries) {

this.burgers = numBurgers;

this.fries = numFries;

}

calculateTotal() {

return (this.burgers \* this.#burgerPrice) + (this.fries \* this.#friesPrice);

}

get totalCost() {

return this.calculateTotal();

}

}

const order1 = new Order(2, 3);

const order2 = new Order(1, 0);

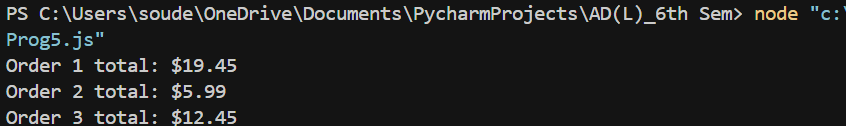
const order3 = new Order(0, 5);

console.log(`Order 1 total: $${order1.totalCost.toFixed(2)}`);

console.log(`Order 2 total: $${order2.totalCost.toFixed(2)}`);

console.log(`Order 3 total: $${order3.totalCost.toFixed(2)}`);

**Output**

*Terminal*