1 You are building a counter application that tracks the number of times a button is clicked. Implement the counter using closure.

Solution:

```
function createCounter () {
 let count = 0;

 return function () {
  count++;
  console.log(`Click count : ${count}`);
 };
}

let clickCounter = createCounter ();

// Simulate button clicks
clickCounter (); // Output: Click count: 1
clickCounter (); // Output: Click count: 2
```

2 You have an object representing a customer order with properties orderId, productName, and quantity. Use destructuring to extract and print these properties.

```
const cart = shoppingCart();

console.log('Cart Items:', cart.getCartItems());

// OUTPUT: Cart Items: []
```

Solution:

```
let order = {
 orderid: "123456",
 productName: "Laptop",
 quantity: 2
};

// Without destructuring
console.log("Without Destructuring:");
console.log( "Order ID:", order.orderid) ;
console.log ("Product Name:", order.productName) ;
console.log( "Quantity:", order.quantity) ;

// With destructuring
console.log("\nWith Destructuring:");
let {orderid, productName, quantity }=order;
console.log ("Order ID:", orderid);
console.log( "Product Name:", productName) ;
console.log( "Quantity:", quantity );
```

3. In this coding challenge let's try to implement the cart feature using javascript closure. Using JS closures try to create a cart array and return a function to getCartItems.

```
const cart = shoppingCart();

console.log('Cart Items:', cart.getCartItems());

// OUTPUT: Cart Items: []
```

Solution:

```
function shoppingCart() {
  canst cartItems = [];
  return {

    getCartitems : function () {
      return cartitems;
    }
  };
}

canst cart = shoppingCart();

console.log('Cart Items:', cart.getCartitems());

// OUTPUT: Cart Items: []
```

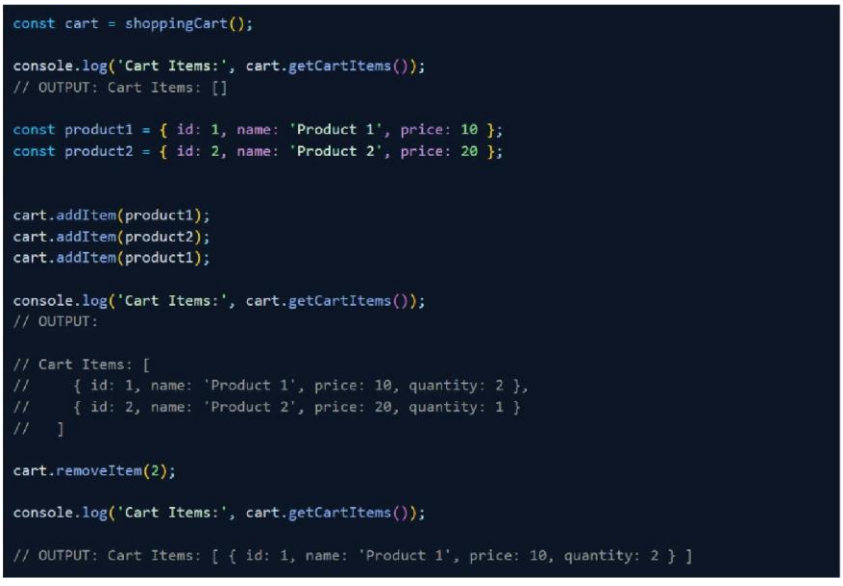
4. Continuing the previous coding challenge, now let's implement the add to cart feature. On calling add to cart closure function, the object of productId, name, quantity and price should be added to the cartItem. Note that if duplicate items with same prouductId is added, the product quantity must be incremented. Use javascript closures to achieve the output.

```
const cart = shoppingCart();

console.log('Cart Items:', cart.getCartItems());
// OUTPUT: Cart Items: []


const product1 = { id: 1, name: 'Product 1', price: 10 };
const product2 = { id: 2, name: 'Product 2', price: 20 };


cart.addItem(product1);
cart.addItem(product2);
cart.addItem(product1);

console.log('Cart Items:', cart.getCartItems());

// OUTPUT:

// Cart Items: [
//     { id: 1, name: 'Product 1', price: 10, quantity: 2 },
//     { id: 2, name: 'Product 2', price: 20, quantity: 1 }
//   ]
```

Solution:

```
function shoppingCart() {
  const cartItems = [];

  function itemindexinCart(productid) {
    return cartitems.findindex(itern => itern.id === productid);
  }

  return {
    additem: function (product) {
      const itemindex = itemindexinCart(product.id);

      if (itemindex !== -1) {
        cartitems[itemindex].guantitx++;
      } else {
        cartitems.push({ id: product.id, name: product.name, price: product.price,
quantity: 1 });
      }
    },

    getCartitems: function () {
      return cartItems;
    }
  };
}

const cart = shoppingCart();

console.log('Cart Items:', cart.getCartitems());
// OUTPUT: Cart Items: []

const product1 = { id: 1, name: 'Product 1', price: 10 };
const product2 = { id: 2, name: 'Product 2', price: 20 };

cart.additem(productl);
cart.additem(product2);
cart.additem(productl);

console.log('Cart Items:', cart.getCartitems());

// OUTPUT:

// Cart Items:
//       id: 1. name: 'Product 1. price: 10. quantity: 2 },
//     { id: 2, name: 'Product 2' price: 20, quantity:  }
//
```

5. Continuing the previous coding challenge, now let's implement the remove item from cart feature. On calling the remove item closure function, the specified productId item must be removed from cartItems array.Usejavascri pt closures to achieve the output.



```
const cart = shoppingCart();

console.log('Cart Items:', cart.getCartItems());
// OUTPUT: Cart Items: []

const product1 = { id: 1, name: 'Product 1', price: 10 };
const product2 = { id: 2, name: 'Product 2', price: 20 };


cart.addItem(product1);
cart.addItem(product2);
cart.addItem(product1);

console.log('Cart Items:', cart.getCartItems());
// OUTPUT:

// Cart Items: [
//     { id: 1, name: 'Product 1', price: 10, quantity: 2 },
//     { id: 2, name: 'Product 2', price: 20, quantity: 1 }
//   ]

cart.removeItem(2);

console.log('Cart Items:', cart.getCartItems());

// OUTPUT: Cart Items: [ { id: 1, name: 'Product 1', price: 10, quantity: 2 } ]
```

Solution:

```
function shoppingCart() {
  const cartItems = [];

  function itemIndexInCart(productId) {
    return cartItems.findIndex(item => item.id === productId);
  }

  return {
    addItem: function(product) {
      const itemIndex = itemIndexInCart(product.id);

      if (itemIndex !== -1) {
        cartItems[itemIndex].
      } else {
        cartItems.push({ id: product.id, name: product.name, price: product.price,
quantity: 1 });
      }
```

```
    },

    removeitem: function (productid) {
      const indexToRemove = itemindexinCart (productId) :

      if (indexToRemove !== -1 ) {
        cartitems .splice (indexTaRemave, 1 );
      }
    },

    getCartitems: function (){
      return cartItems;
    }
  };


canst cart = shoppingCart ():

cansale.lag("Cart Items:', cart.getCartitems ());
JI OUTPUT: Cart Items: []

canst productl = { id: 1, name : ·Product 1 · price: 10 };
const product2 "{id:2, name : ·Product 2 · price :20 };


cart.additem{product 1);
cart.additern(product2) ;
cart.additem( product 1);

console.log("Cart Items:', cart.getCartitems ());
II OUTPUT:

const product1 ={ id: 1, name : ·Product 1 ·, price: 10 };
canst product2 = {id:2, name : 'Product 2', price: 20};


cart.additem(prod uctl);
cart.additem( product2) ;
cart.additem(prod uct1);

console.log( 'Cart Items: ', cart.getCartitems ()):
II OUTPUT:

// Cart Items:
//         id: 1• name: ·Product 1'' price: 10, quantity: 2 }'
//      { id: 2, name: 'Product 2'' price: 20, quantity:  }
JI

cart.removeitem( 2);

console.log( 'Cart Items: ', cart.getCartltems ()):

JI OUTPUT: Cart Items: [ { id: 1, name: 'Product 1', price: 10, quantity: 2 } ]
```

6. You are developing a music playlist management system. Implement functions that leverage closures and higher-order functions to perform common playlist operations.

Task 1: Create a function createPlaylist that takes a playlist name as a parameter and returns a closure. This closure should allow adding and listing songs for the given playlist.

Task 2: Create a function addSong that takes a song name and artist as parameters and adds the song to the specified playlist. Use the closure created in TASK 1.

Task 3: Create a function listSongs that lists all the songs in a specified playlist. Use the closure created in the Task I

```
// Task: Playlist Management
const myPlaylist = createPlaylist("My Favorites");

addSong(myPlaylist, "Song1", "Artist1");
addSong(myPlaylist, "Song2", "Artist2");
addSong(myPlaylist, "Song3", "Artist3");

listSongs(myPlaylist); // Output: My Favorites Playlist: Song1 by Artist1, Song2 by Artist2, Song3 by Artist3
```

Solution:

```
// Task 1: Create Playlist
function createPlaylist(playlistName) {
 let playlist = [];

 return {
  addSong: function(songName, artist) {
   playlist.push({ songNiame, artist });
  },
  listSongs : function ( ) {
   if (playlist.lengtll :=== 0) {
    console.log("${playlistName} Playlist is empty.·);
   }else {
    console.log("${playlistName} Playlist : ${playlist.map(song >
·${song.songName} by ${song.artist}·).join(·, ·)}·);
   }
```

```
  }
 };
}

// Task 2: Add Song to Playlist
function addSong(playlist, songName, artist) {
 playlist.addSong(songName, artist);
}

// Task 3: List Songs in Playlist
function listSongs(playlist) {
 playlist.listSongs();
}


const myPlaylist ;;; createPlaylist("My Favorites");

addSong(myPlaylist, "Sorig1", "Artist1");
addSong(myPlaylist, "Song2", "Artist2");
addSong(myPlaylist, "Song3", "Artist3");

listSongs(myPlaylist); // Output: My avorites Playlist: Song1 by Artist1, Song2
by Artist2,          by Artist3
```