Module 1.1 and 1.2 (for students who retake module 1.1 and 1.2) (30 Points)

Order Class

Properties:

- **orderId**: A **numeric** identifier for the order.
- **customerName**: A **string** representing the name of the customer who placed the order.
- **items**: An **array** that stores the items in the order. Each item is an object consisting of a name, price, and quantity.

Methods:

- **constructor(orderId, customerName):** Initializes a new order with the specified property: orderId and property: customerName, and sets up an empty array items list to items property.

For example, const myOrder = new Order(1001, "Jonathan Wick"), the items in order 1001 must be started empty.

- **addItem(item):** Before adding, verifies an item object is valid: it's not null or undefined, has a non-empty item name and both price and quantity are more than 0. Adds the item to items property and return new item; if not valid, returns -1.

```
myOrder.addItem({ name: "Laptop", price: 1000, quantity: 1 }); // Adds Laptop and return this item
myOrder.addItem({ name: "Keyboard", price: 100, quantity: 2 }); // Adds Keyboard and return this item
myOrder.addItem({ name: "Mouse", price: 25, quantity: 3 }); // Adds Mouse and return this item
myOrder.addItem({ name: "", price: 100, quantity: 1 }); // Empty name, should not add and return -1
myOrder.addItem({ name: "Headphones", price: -50, quantity: 1 }); // Negative price, should not add and return -1
```

- **removeItem(itemName):** Removes item from the items property based on its name, case insensitive, and updates items accordingly. No value returned.

```
myOrder.removeItem("Laptop"); // Removes Laptop if it exists
myOrder.removeItem("Keyboard"); // Removes Keyboard if it exists
myOrder.removeItem("NonExistentItem"); // Tries to remove a non-existent item
myOrder.removeItem(""); // Tries to remove an item with an empty name
myOrder.removeItem("Mouse"); // Removes Mouse if it exists
```

- **calculateTotal()**: Returns the total price for items property by summing the product of the price and quantity of each item.

```
myOrder.calculateTotal(); // Calculates total price
```

- **getDiscountedTotal(discountPercentage)**: Returns the total price after applying a given discount percentage in a range 1-100%. If not, return string “invalid discount percentage”

```
myOrder.getDiscountedTotal(10); // Applies 10% discount if total price is 100 will return 90
myOrder.getDiscountedTotal(0); // No discount applied (0%) return “invalid discount percentage”
myOrder.getDiscountedTotal(100); // Applies 100% discount if total price is 100 will return 0
myOrder.getDiscountedTotal(-10); // Negative discount percentage, should not apply return “invalid discount percentage”
myOrder.getDiscountedTotal(150); // Discount more than 100%, should not apply return “invalid discount percentage”
```

- **findItemByName(itemName)**: Returns the first matching item by name from a collection, or null if none match.

```
myOrder.findItemByName("Laptop"); // Finds Laptop if it exists
myOrder.findItemByName("Keyboard"); // Finds Keyboard if it exists
myOrder.findItemByName("NonExistentItem"); // Tries to find a non-existent item return null
myOrder.findItemByName(""); // Tries to find an item with an empty name return null
```

For retake Module 1.1, sortItemsByPrice(sortingOrder) and isItem(Item), you choose and complete at least one method, you can do both but must be completely.

- **sortItemsByPrice(sortingOrder)**: Sorts items property by price in 'ascending' or 'descending' order and returns a new sorted array. In case invalid sorting order or empty items, return empty array []. Otherwise, it returns a new sorted array and does not modify the original **items** array in the Order class.

```
myOrder.sortItemsByPrice("ascending") // return new array with ascending
myOrder.sortItemsByPrice("descending") // return new array with descending
myOrder.sortItemsByPrice("nonexistentOrder") // return empty array if invalid sorting order
myOrder.sortItemsByPrice("ascending") // return empty array if items array is empty
```

- isItem(Item): Item is an object consisting of a name, price, and quantity. Verifies an item object is valid: it's not null or undefined, has a non-empty item name, and both price and quantity are more than 0. Returns **true** if valid, **false** otherwise.

```
myOrder.isItem({ name: "Pen", price: 3, quantity: 10 }) // valid item return true
myOrder.isItem({ name: "Paper", price: 5, quantity: 20 }) // valid item return true
myOrder.isItem({ price: 20, quantity: 1 }) // missing name return false
myOrder.isItem({ name: "Notebook", price: -10, quantity: 5 }) // negative price return false
myOrder.isItem({ name: "Notebook", price: 10, quantity: -5 }) // negative quantity return false
myOrder.isItem({ name: "", price: 15, quantity: 5 }) // empty name return false
```