Name:_____Student #: _____Section:___.

BTI225 – A2

[40 marks]

Due: Tuesday, Feb 11th, 2020 @ 23:59

Objective:

- Practicing String, Array, user-defined objects, array of objects
- Working with Data.

Specifications:

Given an array of objects with a strictly formatted set of sample data (in the provided file **a2.js)**, serving as "database", create a user-defined object called **orderDB** based on the following specifications. Test this object **orderDB** using the given testing statements (in the section of "**TEST DATA**" in the provided file **a2.js).** A complete result/output is available in file **a2_output.pdf** for your reference. You may need your own necessary testing statements to accomplish this task.

You'll write your code in the file **a2.js**, which includes two blocks of code:

1) data, array of objects (**var allData**)
2) testing statements (/* TEST DATA */)

- you will write code to construct **orderDB** object based on the specifications.
- The block labeled **TEST DATA** at the bottom is currently commented out, but you must uncomment it gradually to test your solution step by step. After running your code with the **TEST Data** uncommented, the output in the web console should look like the sample file (***a2_output.pdf***). In this document, whenever the term "output" is used, that's output to the web console with **console.log().**

- The block of code at the top of the file (commented as /*** **ALL DATA ***/**) is an array of objects named "**allData**". All the objects in this array follow the same format:

{ type: "customer" / or "order" or "product"*/, data: { /* object with properties related to the "type" */ }*

The "**type**" is one of "**customer**", "**order**", or "**product**", and the "**data**" is an object with properties that belong to the "**customer**", "**order**", or "**product**". The properties in "**data**" object are explained as follows for each "type".

type: "customer"	
customerId	A primary key used to uniquely identify each customer.
firstName	The customer's first name.
lastName	The customer's last name.
email	The customer's email address.

type: "order"	
orderId	A primary key that uniquely identifies the order.
customerId	A foreign key identifying the customer who placed this order
orderDate	The date of the order is placed.

type: "product"	
productId	A primary key used to uniquely identify each product.
orderId	A foreign key identifying the order with this product
amount	The amount of the product in the order
price	The price of the product

You can imagine it's a customer-order-product system. The relationships are like:

One customer can place many orders.

Each order can include many products.

Construct Object orderDB

In ***a2.js*** file, underneath the "**allData**" array, declare an object called **orderDB** using "Object Literal Notation". This object will **contain** the following **properties** & **methods** that need you to complete based on the following specifications.

1. Properties (/attributes/data): [3 marks]

The following are internal arrays that will serve as the primary storage mechanism within the **orderDB** object for all of the sample data.

1) **customers:**
   This is an array that will contain all the sample data objects that are of **type "customer".** It is initialized as an empty array (ie, []) and will be manipulated using the methods in the Object **OrderDB** (defined below).

2) **orders:**
   This is an array that will contain all the sample data objects that are of **type "order".** It is initialized as an empty array (ie, []) and will be manipulated using the methods in the Object **OrderDB** (define below).

3) **products:**
This is an array that will contain all the sample data objects that are of **type "product".** It is initialized as an empty array (ie, []) and will be manipulated using the methods in the Object **OrderDB** (define below).

2. Methods (/functions/logic):

You are going to define 10 methods (functions) associated with the object **orderDB**. Please label each method with numbers in comment for convenience. e.g.,

// 1) constructArrays(allData)

Note: You can add more regular functions (not associated with the object) if needed.

1) **constructArrays(allData)** **[4 marks]**

It is the first method that will be invoked on your **OrderDB** Object. It is this method that takes all of the sample data (array "**allData**") and inserts it into the correct arrays (ie: "**customers**", "**orders**", or "**products**"). It takes one parameter, the array "**allData**" from the top part of file **a2.js** and processes it into three arrays of objects (i.e., **customers**, **orders**, **products**, which are properties of object **orderDB**) following the rules:

- if type is "**customer**", insert the "**data**" object into the array "**customers**" .
(**HINT**: call function **addCustomer(customer),** which will be defined below)

- if type is "**order**", retrieve the "**data**" object and set its "**orderDate**" property to the current date and add it into the array "**orders**" .
(**HINT**: call function **addOrder(order)**, which will be defined below)
- if type is "**product**", insert the "**data**" object into the array "**products**".
(**HINT**: call function **addProduct(product)** that will be defined below)

Once this method **constructArrays(allData)** has run, your "database" is built. The result will be:

- array "**customers**" should contain **4** "customer" data objects,
- array "**orders**" should contain **10** "order" data objects,
- array "**product**" should contain **16** "product" data objects

HINT: The following methods/ functions also belong to the object **orderDB**. To refer to the property arrays of object **orderDB** (e.g., "**customers**") from within these methods, use "**this**" keyword, i.e.: "**this.customers**".

2) **addCustomer (customer)** **[2 marks]**

This method takes one parameter of object type "**customer**" and adds it to the array "**customers**".

3) **addOrder (order)** **[3 marks]**

- This method takes one parameter of object type "**order**".
- Sets the property **orderDate** of parameter (**order**) to the current date.
- Add object **order** to the array "**orders**".

4) **addProduct (product)** **[2 marks]**

This method takes one parameter of object type "**product**" and adds it to the array "**products**".

5) displayCustomerById(custId) [3 marks]

This method takes a number representing a **customerId** and outputs the customer data for the corresponding **customerId** from the array "**customers**". Display in the following format:

For example, with the testing statement:

orderDB.displayCustomerById(1);

The display/result on web console is as follows.

```
=============Display Customer By Id: 1==================
Customer: 1, First Name: Dave, Last Name: Bennett, E-mail: dbennett@gmail.com.
```

6) displayAllCustomers [2 marks]

This method takes no parameters and simply outputs all **customers** in the **same format as above 5)**, including a header at the top of the output stating "**Display All Customers**". The display/result on web console is as follows.

```
=============Display All Customers==================
Customer: 1, First Name: Dave, Last Name: Bennett, E-mail: dbennett@gmail.com.
Customer: 2, First Name: John, Last Name: Stevens, E-mail: jstevens22@hotmail.com.
Customer: 3, First Name: Sarah, Last Name: Pym, E-mail: spym99@hotmail.com.
Customer: 4, First Name: Steven, Last Name: Edwards, E-mail: steven2231@hotmail.com.
```

7) displayAllOrders( ) [5 marks]

This method takes no parameters and outputs all **orders** in the following format, including a header at the top of the output stating "**Display All Orders**". You need to refer to the array "**customers**" via the foreign key "**customerId**" to get the customers' **first name** and **last name** information. The display/result on web console is as follows.

```
=============Display All Orders==================
OrderId 101 was placed by customer Dave Bennett on the day of 1/29/2020, 2:27:02 PM.
OrderId 102 was placed by customer Dave Bennett on the day of 1/29/2020, 2:27:02 PM.
OrderId 103 was placed by customer John Stevens on the day of 1/29/2020, 2:27:02 PM.
OrderId 104 was placed by customer John Stevens on the day of 1/29/2020, 2:27:02 PM.
OrderId 105 was placed by customer Dave Bennett on the day of 1/29/2020, 2:27:02 PM.
OrderId 106 was placed by customer Dave Bennett on the day of 1/29/2020, 2:27:02 PM.
OrderId 107 was placed by customer Sarah Pym on the day of 1/29/2020, 2:27:02 PM.
OrderId 108 was placed by customer Sarah Pym on the day of 1/29/2020, 2:27:02 PM.
OrderId 109 was placed by customer Sarah Pym on the day of 1/29/2020, 2:27:02 PM.
OrderId 110 was placed by customer Steven Edwards on the day of 1/29/2020, 2:27:02 PM.
```

8) displayAllProducts() [2 marks]

This method takes no parameters and outputs all **products** in the following format, including a header at the top of the output stating "**Display All Products**". The display/result on web console is as follows.

```
============Display All Products==================
Product ID: 100001, Order ID: 105, Amount: 2, Price: $102.5.
Product ID: 100002, Order ID: 101, Amount: 5, Price: $22.45.
Product ID: 100003, Order ID: 104, Amount: 1, Price: $892.55.
Product ID: 100004, Order ID: 104, Amount: 6, Price: $23.32.
Product ID: 100005, Order ID: 102, Amount: 2, Price: $90.67.
Product ID: 100006, Order ID: 106, Amount: 1, Price: $342.39.
Product ID: 100007, Order ID: 109, Amount: 12, Price: $7.32.
Product ID: 100008, Order ID: 102, Amount: 4, Price: $12.12.
Product ID: 100009, Order ID: 103, Amount: 3, Price: $56.32.
Product ID: 100010, Order ID: 107, Amount: 2, Price: $152.38.
Product ID: 100011, Order ID: 108, Amount: 5, Price: $66.32.
Product ID: 100012, Order ID: 110, Amount: 3, Price: $96.92.
Product ID: 100013, Order ID: 108, Amount: 1, Price: $532.78.
Product ID: 100014, Order ID: 102, Amount: 6, Price: $36.32.
Product ID: 100015, Order ID: 109, Amount: 3, Price: $56.32.
Product ID: 100016, Order ID: 108, Amount: 1, Price: $366.32.
```

9) displaySales() **[6 marks]**

This method searches all the arrays to collect the order information for all customers. The display format is as follows:

Customer <firstName lastName> placed <number of orders> orders. Total cost is $<total cost for all the orders>

Where the above information in <…> is retrieved from the arrays, i.e.: **customers**, **orders**, and **products**.

Hint: the **cost** for each product is **amount** * **price** from the array **product**. The **total cost** is total **costs** of all the **products** of **all orders** for the specific **customer**.

The display/result is as follows.

```
============Display All Customers' Order Info==================
Customer Dave Bennett placed 4 orders. Total cost is $1107.38.
Customer John Stevens placed 2 orders. Total cost is $1201.43.
Customer Sarah Pym placed 3 orders. Total cost is $1792.26.
Customer Steven Edwards placed 1 orders. Total cost is $290.76.
```

10) removeOrderById(orderId) **[8 marks]**

This method takes a number representing an **orderId** and searches through the **orders** array and removes the **order** with the matching "**orderId**" property from the array. Also, this method must ensure that **all the products with "orderId" are removed** from the array **products**.

Display all orders and all products after the removal.

If **orderId** does not exist in the array **orders**, your program reports:

OrderId <orderId> is not found

Where <orderId> is the orderId you try to remove.

For example, if try to remove orderId 177, it's not found in the array. If remove orderId 101, the order with orderId 101 is removed from array **orders**. Also, the product with productID 100002 was removed from the array **product** because its orderId is 101.
Hint: try function splice()
The result/display on web console is as follows.

```
OrderId 177 is not found.
==== Remove Order ID 101  ===
=============Display All Orders==================
OrderId 102 was placed by customer Dave Bennett on the day of 1/29/2020, 2:27:02 PM.
OrderId 103 was placed by customer John Stevens on the day of 1/29/2020, 2:27:02 PM.
OrderId 104 was placed by customer John Stevens on the day of 1/29/2020, 2:27:02 PM.
OrderId 105 was placed by customer Dave Bennett on the day of 1/29/2020, 2:27:02 PM.
OrderId 106 was placed by customer Dave Bennett on the day of 1/29/2020, 2:27:02 PM.
OrderId 107 was placed by customer Sarah Pym on the day of 1/29/2020, 2:27:02 PM.
OrderId 108 was placed by customer Sarah Pym on the day of 1/29/2020, 2:27:02 PM.
OrderId 109 was placed by customer Sarah Pym on the day of 1/29/2020, 2:27:02 PM.
OrderId 110 was placed by customer Steven Edwards on the day of 1/29/2020, 2:27:02 PM.
=============Display All Products==================
Product ID: 100001, Order ID: 105, Amount: 2, Price: $102.5.
Product ID: 100003, Order ID: 104, Amount: 1, Price: $892.55.
Product ID: 100004, Order ID: 104, Amount: 6, Price: $23.32.
Product ID: 100005, Order ID: 102, Amount: 2, Price: $90.67.
Product ID: 100006, Order ID: 106, Amount: 1, Price: $342.39.
Product ID: 100007, Order ID: 109, Amount: 12, Price: $7.32.
Product ID: 100008, Order ID: 102, Amount: 4, Price: $12.12.
Product ID: 100009, Order ID: 103, Amount: 3, Price: $56.32.
Product ID: 100010, Order ID: 107, Amount: 2, Price: $152.38.
Product ID: 100011, Order ID: 108, Amount: 5, Price: $66.32.
Product ID: 100012, Order ID: 110, Amount: 3, Price: $96.92.
Product ID: 100013, Order ID: 108, Amount: 1, Price: $532.78.
Product ID: 100014, Order ID: 102, Amount: 6, Price: $36.32.
Product ID: 100015, Order ID: 109, Amount: 3, Price: $56.32.
Product ID: 100016, Order ID: 108, Amount: 1, Price: $366.32.
```

TEST DATA Output

Test your program step by step. For example, write a function and test it immediately (if possible) before write next function. The advantage is to avoid the program becomes too complicated and errors may interleave and become hard to debug.

Once you're ready to test your solution, uncomment the code block "**TEST DATA**" line by line and run your program.

A complete testing output is available for your convenience in the file ***a2_output.pdf***.

Note, the dates will be different, as you will run your program at a different time.

Name:_____Student #: _____Section:___.

Submission:

Keep your file name as **a2.js** and submit it on blackboard. Make sure that the code block "**TEST DATA**" is uncommented and that your solution gives the correct output (refer to ***a2_output.pdf***).

- Add the following declaration at the top of your code:

```
/*********************************************************************************
* BTI225 – Assignment 02
* I declare that this assignment is my own work in accordance with Seneca  Academic Policy.
* No part of this assignment has been copied manually or electronically from any other source
* (including web sites) or distributed to other students.
*
* Name: _____ Student ID: _____ Date: _____
*
*********************************************************************************/
```

- Submit your **a2.js** file to My.Seneca under **Assignments** -> **Assignment** 2

- Late submission:

10% penalty each day for up to 5 school days. After that, no submission will be accepted.