National University of Computer and Emerging Sciences
Karachi - Campus

(JavaScript)

Q1: Do All the Following Tasks..... (15 marks)

Open the folder named "JAVASCRIPT". Inside this folder, you will find a file named **Index.js**. Write all your JavaScript code in this file.

You are tasked with managing a list of bank accounts for a banking system. Each account can have 2 products (Debit Cards, Retail Loans). Details for each account and product type are provided below.

Bank Account Details:

Each bank account has the following attributes:

- **Account Title** (String)
- **Account Number** (String)
- **IBAN Number** (String)
- **Account Balance** (Number)
- **Products** (List of Objects)
    - Products can include **Debit Cards** or **Retail Loans**.

Product Details:

1. **Debit Card**
    - **Category** (String): e.g., Platinum, Gold, Silver
    - **Valid Till** (String): e.g., "11/29"
    - **Card Number** (Number)
    - **CVV** (Number)
2. **Retail Loan**
    - **Credit Score** (Number): A higher score allows access to larger loans.
    - **Loan Amount** (Number): Loan amount is added to the **Account Balance** when approved.

Bank Account

Account Title	Account Number	IBAN Number	Account Balance	Products
Fatima Javed	123456789012	PK12ICT12345678912345	3000	
Mitesh Arun	223456789012	PK12ICT12345678912346	5000	Debit Card
Hassan Raza	323456789012	PK12ICT12345678912347	4000	Debit Card, Retail Loan

Debit Card

Category	Valid Till	Card Number	CVV
Platinum	11/29	123456788	123
Gold	09/28	123456789	456

Retail Loan

Credit Score	Loan Amount
500	2000

Task 1

Create the list of bank accounts as described in the table above.

Task 2

Account **223456789012** wants a loan. Their credit score is **70**, and they want to borrow **1000**.

1. If the credit score is between **50 and 100**, the maximum loan they can receive is **3000**.
2. If eligible, add the Retail Loan object to their products and update their account balance by adding the loan amount.
   Write a function to implement this logic.

Task 3

Write a JavaScript function that:

1. Takes an **Account Number** and **Details of a New Product** as input.
2. Finds the account by its number and prints the **Account Title** in the console.
3. Adds the new product to the account, product can either by Debit Card Object or Retail Loan Object

Task 4

Account **323456789012** wants to update their **Account Title** to **Hassan Raza Lakhany.**

- Write a function to check if the account exists and update its title.

Task 5

Account **323456789012** wants to pay off their loan.

1. Check if the account exists and retrieve the loan details.
2. If their **Account Balance** is sufficient to pay off the loan, subtract the loan amount from their balance and remove the Retail Loan object from their products.
3. If the balance is insufficient, display an error message: **"Insufficient balance to pay the loan."**

Task 6

Account **223456789012** wants to upgrade their Debit Card from **Platinum** to **Gold.**

- Write a function to check if the account exists and update the **Category** of their Debit Card.

Task 7

Account **Fatima Javed** wants to close her account.

- Write a function to remove her account from the list of bank accounts.

Task 8

A new customer, **Zumair Shamsi**, wants to open a bank account with a **Debit Card.**

- Append his details to the bank accounts list:

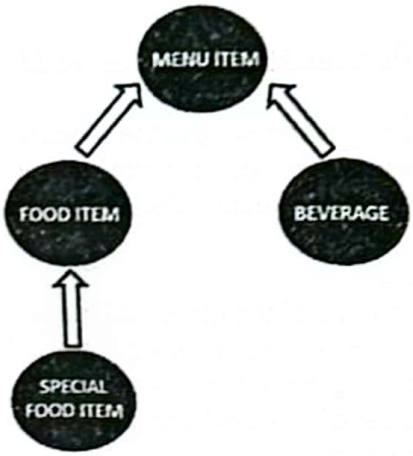
Account Title	Account Number	IBAN Number	Account Balance	Products
Zumair Shamsi	123456789014	PK14ICT12345678912345	10000	Debit Card

Category	Valid Till	Card Number	CVV
Silver	11/29	123456780	123

[OOP]

Q2: Do All the Following Tasks .....[15 marks]

Open the **'OOP'** folder, where you will find the **'Restaurant.cpp'** file. Boilerplate code has been provided for your convenience



TASK 1: Class Structure

MenuItem Class:

- Create a base class named **MenuItem** with the following attributes:
  - Name (String)
  - Price (float)
  - Id (int)

FoodItem Class:

- Create a class **FoodItem** that inherits from **MenuItem**. This class should have an additional attribute:
  - CuisineType (String) – e.g., Chinese, Italian, etc.

Beverage Class:

- Create a class **Beverage** that inherits from **MenuItem**. This class should have an additional attribute:
  - BeverageType (String) – e.g., Juices, Milkshake, Cola, etc.

SpecialFoodItem Class:

- Create a class **SpecialFoodItem** that inherits from the **FoodItem** class. This class should have an additional attribute:
  - SpecialOffer (String) – e.g., "20% off", "Buy 1 Get 1 Free", etc.

TASK #2: Encapsulation

- **Ensure encapsulation** in all classes by:
  - Making all attributes **private**.
  - Providing **public getter (accessor)** and **setter (mutator)** methods for each attribute in all classes.

TASK #3: Constructor Overloading in MenuItem Class

- Implement constructor overloading in the **MenuItem** class:
  - **Default Constructor:** Initializes attributes with default values.
  - **Parameterized Constructor**: Takes id, name, and price as parameters to initialize the object.
- In the main function:
  - Create two objects of the MenuItem class:
    - One using the default constructor.
    - One using the parameterized constructor.
  - Print the values of id, name, and price for both objects to demonstrate the use of both constructors.

TASK #4: Method Overloading in MenuItem Class

- Create three overloaded **describe()** methods in the **MenuItem** class:
  1. **First Version:**
     - Takes no arguments and prints: "This is a menu item."
  2. **Second Version:**
     - Takes an argument extraInfo (String) and prints: "This is a menu item: [extraInfo]."
  3. **Third Version**:
     - Takes an argument extraInfo (String) and returns it as a string.
- In the main function:
  - Create a MenuItem object.
  - Call each version of the describe() method and describe its behavior in comments.

TASK #5: Displaying Special Food Item Details

- In the **SpecialFoodItem** class, create a function displayDetails() that prints all the attributes inherited from the MenuItem and FoodItem classes, as well as its own attribute (specialOffer).

  Example Output:

  Special Food Item ID: [id]
  Special Food Item Name: [name]
  Special Food Item Price: [price]
  Cuisine Type: [cuisineType]
  Special Offer: [specialOffer]

- In the main function:
  - Create an object of the SpecialFoodItem class.
  - Use the setter methods to set values for all attributes.
  - Call the displayDetails() function to display the details of the special food item.

[HTML-CSS]

Q3: Do the Following ..... [10 marks]

Open the folder named **"HTML-CSS"**.

1. Within this folder, you will find the following files:
   a. **index.html** (for the HTML structure)
   b. **style.css** (for styling the webpage)
2. Your task is to:
   a. Write the necessary HTML code in **index.html**.
   b. Write the required CSS code in **style.css**.
3. Use the provided **Assets** folder for the images:
   a. Picture1.png
   b. Picture2.png
4. 'Achieve the layout and design as displayed in **Output.png**.
5. Use the following color codes in your design:
   a. **Darker Color:** #f2812a
   b. **Lighter Color:** #d9d9d9

[MCQS]

Q4: Do the Following ..... [10 marks]

Open the folder named **"MCQS"**. Inside this folder, you will find a file named **MCQS.doc** containing **12 multiple-choice questions.**

1. For the option you believe is correct, change the text color of the entire option to **RED.**
2. A sample has been provided at the top of the **MCQS.doc** file to guide you.