**PRACTICAL EXAMINATION**

**Subject code: PRN212 - SU25**  
**Duration: 85 minutes**

**Question 1 (3.5 marks):**

**1. Create a Console Project with name LibraryRecordApp (0.5 mark)**

**2. Create a Generic Class (0.5 mark):**

* Create a generic class called **BookCollection<T>** that can store library-related data of any type.
* The class should have:
  + A property **Items** of type **List<T>** to store a collection of library items.
  + A method **AddItem(T item)** to add an item to the list.
  + A method **GetAllItems()** to return an **IEnumerable<T>** containing all items.
  + A method **RemoveItem(T item)** to remove a specific item from the list.
  + A method **GetItemCount()** to return the total number of items.

**3. Create a Book class (1.0 mark):**

* The **Book** class should have properties: **BookID** (int), **Title** (string), **Author** (string), **PublishedYear** (int), **Price** (decimal).
* **GetBookAge():** Returns book age as int (current year - published year).
* Override the **ToString()** method to display book information in the format:  
  **"BookID - Title by Author (PublishedYear) - $Price"**

**4. Use the Generic Class (1.5 mark):**

In the Main function, you must implement the following tasks:

* Create an instance of **BookCollection<string>** and add 3 book categories using **AddItem()** method.
* Display all categories and the total count using **GetAllItems()** and **GetItemCount()** methods.
* Create another instance of **BookCollection<Book>** and add 4 Book objects using **AddItem()** method.
* Use **LINQ** to filter and display books with **price greater than $20**.
* Display books grouped by **book age** using LINQ GroupBy (group by: "New" for age ≤ 5, "Classic" for age > 5).

**Question 2 (6.5 marks):**

**CityLibrary** wants to develop a library management system to track information about books, members, and borrowing records. The specific requirements are as follows:

**1. Create a Library Project named CityLibraryDataAccess within a solution named CityLibrarySystem (1.0 mark):**

Build an application using Entity Framework Core. Create a database using either the **Code First or Database First** approach with the following classes and their relationships:

* **Member.cs** (contains: MemberID, MemberCode, FullName, Email, Password)
* **Book.cs** (contains: BookID, ISBN, Title, Author, AvailableCopies)
* **BorrowRecord.cs** (contains: RecordID, MemberID, BookID, BorrowDate, ReturnDate, Status)

**Notes:** Borrow statuses include: "Borrowed", "Returned" and "Overdue"

**Relationships:**

* One **Member** can have multiple **BorrowRecords** (One-to-Many).
* One **Book** can have multiple **BorrowRecords** (One-to-Many).

**2. Create database DBCityLibrary and Build relational tables (0.5 mark)**

**Notes:** The tables must be related as shown in the requirements above, and these relationships are automatically generated through Code First or Database First.

**3. Create a Library project named CityLibraryBusiness (0.5 Mark)**

**Notes:** CityLibraryDataAccess should be containing any EF, DBContext, DAL, Model classes. CityLibraryBusiness should be containing all application business logics.

**4. Create a WPF application named CityLibraryApp (4.5 Mark)**

**Implement Member Login System (1.0 mark)**

* **Create a login form** where members enter their **MemberCode** and **Password**.
* **Validate credentials** by checking the database.
  + If login is **successful**, redirect to the **MainWindow Form**.
  + If login **fails**, show error message: **"Invalid MemberCode or Password"**.

**MainWindow with Menu (0.5 mark)**

* After successful login, navigate to **MainWindow** containing menu with:
  + Book Management
  + My Borrowings

**Book Management (1 mark)**

* **Display all books** with their basic information (ISBN, Title, Author, Available Copies).
* **Add new books** to the system.
* **Edit book information**.

**My Borrowings (1.5 marks)**

* **Display member's borrow records** with:
  + Book information (ISBN, Title, Author)
  + Borrow Date, Return Date
  + Status (Borrowed, Returned, Overdue)
* **Allow members to update** record status (Return or Mark Overdue).
* **Sort records** ascending by borrow date.
* **Create new borrow records** for books.

**Logout Functionality (0.5 mark)**

* **Logout button** that returns user to login screen.

**Important Notes:**

**Functionality Requirement:** Your submission must be fully functional. Projects that fail to compile or run will receive a score of zero. Ensure all features are correctly implemented and thoroughly tested before submission.

**Submission Requirements:**

* Submit all project files and **backup of the SQL Server database script** (\*.sql) – if you use DatabaseFirst approach.
* Ensure that no **personal identifiers** (name or student ID) are included in the submission.

**Additional Instructions:**

* You may use **local resources**, but internet access is restricted to **downloading the exam, submitting your work, and installing packages via NuGet**.
* Fully implemented according to software architecture and design principles, transition from tight coupling to loose coupling such as **BusinessLogic, DataAccess**.