**Subject: PRF192 - Programming Fundamental with C**

**Assignment**

**Objectives:**

* Gain experience in managing structured data using binary files in C.
* Implement a modular and menu-driven program to handle CRUD (Create, Read, Update, Delete) operations for product data.
* Use struct to represent product information, including product\_id, product\_name, price, and quantity.
* Demonstrate reading, writing, and updating data in a binary file efficiently.
* Enhance problem-solving skills by implementing practical applications of file I/O operations.

**Problem Description:**

Create a product management system where product data, including:

* **Product ID** (unique identifier for each product),
* **Product Name** (name of the product),
* **Price** (unit price of the product),
* **Quantity** (available stock of the product),

is stored in a binary file. The system should allow users to perform the following operations through a menu-driven interface:

1. Write new product data to the file (overwrite existing data).
2. Append additional product records to the file.
3. Read and display all product records.
4. Modify a specific product by searching with its **Product ID**.

The program will simulate a real-world inventory system and include robust error handling for file operations.

**Situation Description:**

Inventory management is a critical component of retail and wholesale businesses. This case study simulates such a system by managing product data using binary files, ensuring efficient storage and retrieval. By incorporating real-world functionality, this program teaches students how to:

* Use binary files for persistent storage,
* Work with structures to group related data fields,
* Develop modular and reusable code,
* Implement CRUD operations effectively.

**Syntax Use in the Problem:**

1. **File Operations:**

* fopen, fread, fwrite, fseek, and fclose to handle binary file data.

1. **Structures**:

* Represent product data using struct Product containing fields for ID, name, price, and quantity.

1. **Menu-Driven Programming**:

* Use a do-while loop and switch statement to implement user interaction.

1. **Modular Functions**:

* Separate functions for writing, appending, reading, and modifying products to enhance code reusability and readability.

1. **String Handling**:

* Use character arrays to store and process the **product\_name**.

**Specific Requirements:**

1. **Product Structure**:

* Include the following fields:
  + product\_id (integer),
  + product\_name (character array with a maximum length of 50),
  + price (float),
  + quantity (integer).

1. **Menu Options**:

* Write products to the file (overwrite existing records).
* Append additional products to the file.
* Read and display all product records.
* Modify a product by its **product\_id**.

1. **File Management**:

* Data is stored in a binary file (products.bin).
* Ensure robust error handling for file operations.

1. **Modular Design**:

* Use separate functions for each operation.

1. **Output Formatting**:

* Display product details in a tabular format when reading data.

**Detailed Evaluation Criteria:**

**Stage 1: Understanding Basics (1.5 points)**

* **Focus:** Design struct Product and implement basic binary file operations.
* **Criteria:**
  + Correct implementation of struct Product with all fields.
  + Proper use of file operations (fopen, fwrite, fclose).
  + Program compiles and runs without errors.

**Stage 2: Menu-Driven Program (1.5 points)**

* **Focus:** Create an interactive menu system.
* **Criteria:**
  + Functional menu using do-while and switch.
  + Handles user inputs effectively.
  + Smooth navigation between menu options.

**Stage 3: Writing and Appending Data (2 points)**

* **Focus:** Implement “Write Products” and “Append Products” operations.
* **Criteria:**
  + “Write Products” correctly overwrites existing data.
  + “Append Products” adds records without overwriting.
  + Data is accurately saved in the binary file.

**Stage 4: Reading Data (1.5 points)**

* **Focus:** Display product records in a tabular format.
* **Criteria:**
  + Reads all product records accurately from the file.
  + Output is clear, formatted, and aligned in a table.

**Stage 5: Modifying Records (1.5 points)**

* **Focus:** Update specific records based on product\_id.
* **Criteria:**
  + Efficiently searches for records by product\_id.
  + Correctly updates fields and saves changes back to the file.
  + Handles cases where the record is not found.

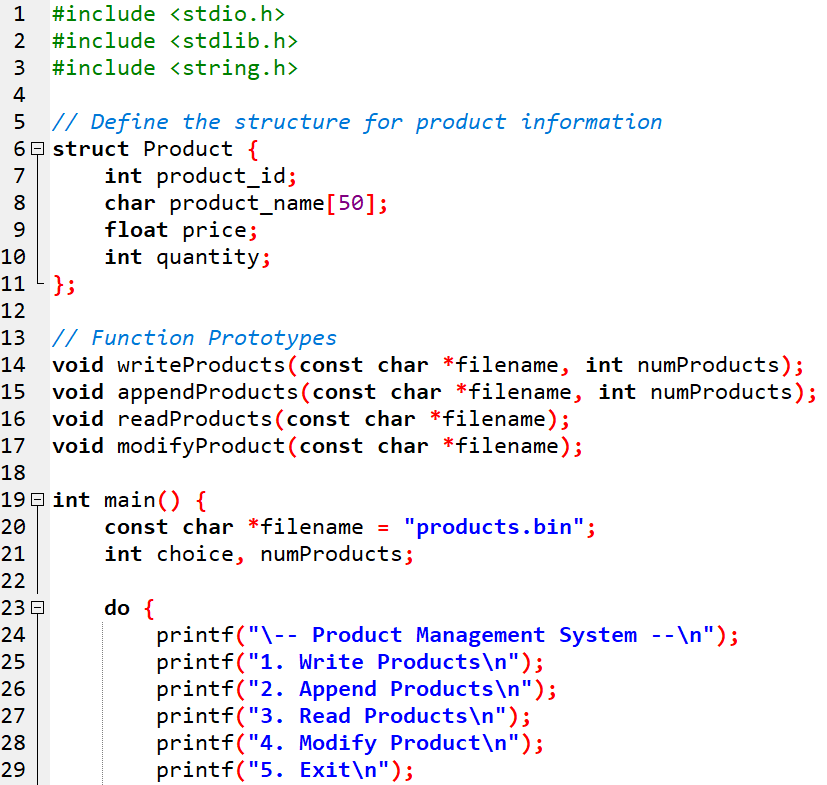
**Stage 6: Error Handling and Robustness (1 point)**

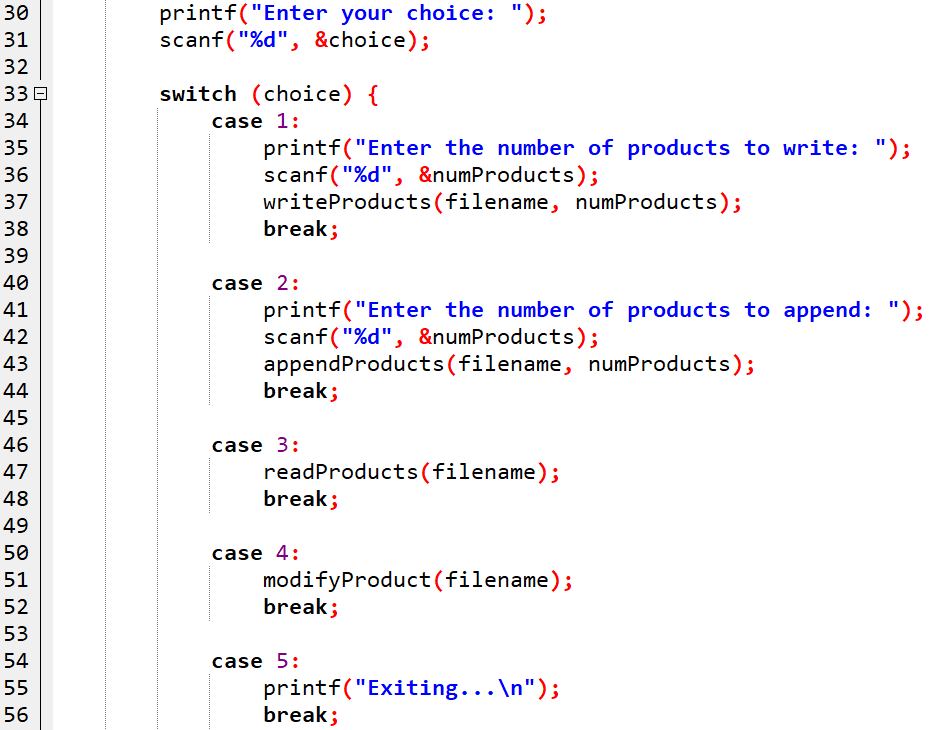
* **Focus:** Prevent invalid inputs and handle file-related errors.
* **Criteria:**
  + Gracefully manages missing or corrupted files.
  + Validates user inputs (e.g., unique product\_id, proper data types).
  + Prevents crashes due to invalid operations.

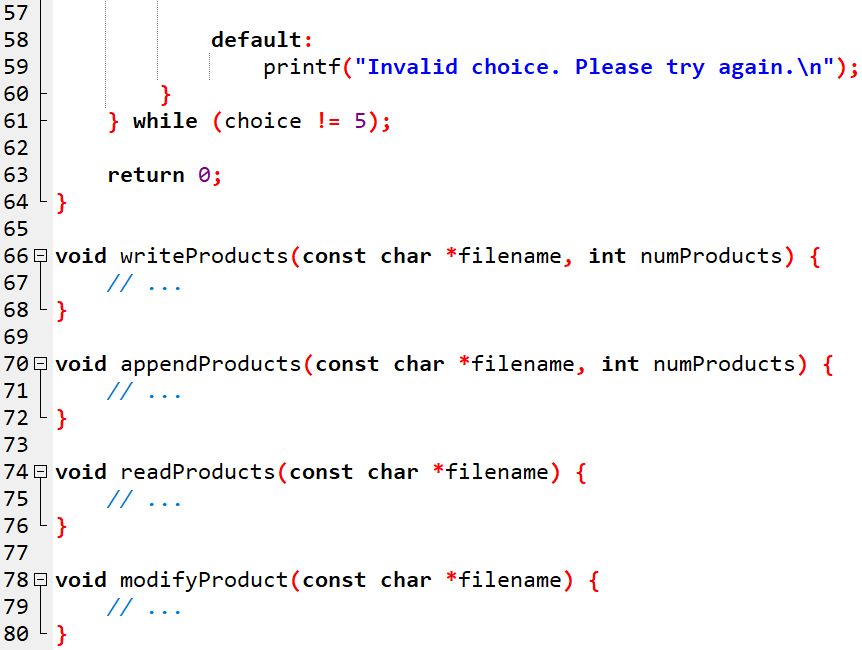
**Stage 7: Final Testing and Optimization (1 point)**

* **Focus:** Debug, test, and optimize the program.
* **Criteria:**
  + All menu options function as intended.
  + Code is clean, readable, and reusable.
  + Thorough testing ensures reliability.

**Code Sample:**







**Output Sample:**

|  |
| --- |
| -- Product Management System --  1. Write Products  2. Append Products  3. Read Products  4. Modify Product  5. Exit  Enter your choice: 1  Enter the number of products to write: 2  Enter details for product 1:  Product ID: 101  Product Name: Laptop  Price: 750.00  Quantity: 15  Enter details for product 2:  Product ID: 102  Product Name: Smartphone  Price: 500.00  Quantity: 30  Products have been written to the file successfully.  -- Product Management System –  1. Write Products  2. Append Products  3. Read Products  4. Modify Product  5. Exit  Enter your choice: 2  Enter the number of products to append: 1  Enter details for product 1:  Product ID: 103  Product Name: Tablet  Price: 300.00  Quantity: 25  Products have been appended to the file successfully.  -- Product Management System --  1. Write Products  2. Append Products  3. Read Products  4. Modify Product  5. Exit  Enter your choice: 3  Reading products from the file:  -----------------------------------------------------  Product ID Product Name Price Quantity  -----------------------------------------------------  101 Laptop 750.00 15  102 Smartphone 500.00 30  103 Tablet 300.00 25  -----------------------------------------------------  -- Product Management System --  1. Write Products  2. Append Products  3. Read Products  4. Modify Product  5. Exit  Enter your choice: 4  Enter the Product ID to modify: 101  Product found. Enter new details:  New Product Name: Gaming Laptop  New Price: 900.00  New Quantity: 10  Product updated successfully.  -- Product Management System --  1. Write Products  2. Append Products  3. Read Products  4. Modify Product  5. Exit  Enter your choice: 5  Exiting... |

**-- End --**