Department of Computer and Software Engineering – ITU

CE101T: Object Oriented Programming

Course Instructor: Nadir Abbas	**Dated: 26/01/2025**
Teaching Assistant: Zainab, Nahal, Bilal	**Semester: Spring 2025**
Session: 2024-2028	**Batch: BSCE2024**

Assignment 1. Utilizing Structs and Nested Structs for Data Modeling

Name	Roll number	Obtained Marks/35

Checked on: _____

Signature: _____

Submission:

- Email instructor or TA if there are any questions. You cannot look at others' solutions or use others' solutions, however, you can discuss it with each other. Plagiarism will be dealt with according to the course policy.
- Submission after due time will not be accepted.

In this assignment you have to do following tasks:

Task 1: Ensure that you have installed all three softwares in your personal computer (Github, Cygwin & CLion). Now, accept the assignment posted in classroom (e.g Google, LMS etc) and after accepting, clone the repository to your computer. Make sure you have logged into the github app with your account.

Task 2: Open Cygwin app, Move to your code directory with following command "cd <path_of_folder>"
<path_of_folder> can be automatically populated by dragging the folder and dropping it to the cygwin window.
Run the code through Cygwin, use command "make run", to get the output of the code

Task 3: Solve the given problems, write code using **CLion** or any other IDE.
Task 4: Keep your code in the respective git cloned folder.
Task 5: Commit and Push the changes through the Github App
Task 5: Write the code in separate files **(as instructed)**. Ensure that file names are in lowercase (e,g **main.cpp**).

Task 6: Run '**make run**' to run C++ code
Task 7: Run '**make test**' to test the C++ code

Problem Statement: You will create an inventory management system for storing and managing products and their associated suppliers. Follow the questions step-by-step to build the program incrementally.

Question 1: Define the Structs

Define the following structs with their variables and data types:

- Supplier: Contains details about a supplier.
  - **supplierName** (string): The name of the supplier.
  - **contactNumber** (string): The supplier's contact number.
- Product: Contains details about a product.
  - **productName** (string): The name of the product.
  - **productPrice** (float): The price of the product.
  - **supplier** (Supplier): A nested struct instance representing the supplier of the product.
- Inventory: Represents the collection of products in the inventory.
  - **products** (Product*): A pointer to a dynamic array of Product.
  - **size** (int): The maximum number of products the inventory can hold.
  - **currentCount** (int): The current number of products in the inventory.

Question 2: Initialize the Inventory

Write a function that initializes an Inventory object. The function should:

- Allocate memory for a dynamic array of Product based on the size passed by the user.
- Set the currentCount to 0.

Function Prototype:
void initializeInventory(Inventory &inv, int size);

Note:
This function should dynamically allocate memory for the products array and set the size and currentCount values appropriately.

Question 3: Set Supplier Data

Write a function that sets the data for a Supplier struct.

- The function should take a Supplier object and set its supplierName and contactNumber.

Function Prototype**:**
void setSupplierData(Supplier &supplier, string supplierName, string contactNumber);

Note:
The function should not handle user input/output but instead assign values passed as arguments to the supplier object.

Question 4: Set Product Data

Write a function that sets the data for a Product struct.

- The function should take a Product object, a Supplier object, and set the productName, productPrice, and supplier fields.

Function Prototype:
void setProductData(Product &product, Supplier supplier, string productName, float productPrice);

Note:
The function should initialize a product's details, including assigning an existing Supplier to the supplier field of the Product.

Question 5: Add a Product to Inventory

Write a function to add a product to the inventory.

- The function should check if the inventory is full.
- If there is space, the function should add the product to the array and increment currentCount.
- If the inventory is full, the function should return 1. Otherwise, it should return 0.

Function Prototype:
int addProductToInventory(Inventory &inv, Product product);

Note:
The function must use the currentCount and size variables in the Inventory struct to manage the addition.

Question 6: Search for a Product by Name

Write a function to search for a product in the inventory by its name.

- The function should iterate through the products array in the inventory.
- If a product with the given name exists, return a pointer to the Supplier of that product.
- If no product is found, return nullptr.

Function Prototype:
Supplier* searchProductByName(Inventory &inv, string productName);

Note:
This function will help in finding details about a specific product's supplier.

Question 7: Display a Product

Write a function that displays the details of a Product.

- It should display the productName, productPrice, supplierName, and contactNumber.

Function Prototype:
void displayProduct(Product &product);

Note:
This function will be used later to display individual product details when called.

Question 8: Display Inventory

Write a function that displays all the products in an inventory.

- The function should iterate through the dynamic array of Product and call displayProduct for each product.
- If the inventory is empty, it should display an appropriate message.

Function Prototype:
void displayInventory(Inventory &inv);

Note:
Use the currentCount in the Inventory struct to determine how many products to display.

Assessment Rubric for Assignment

Performance metric	CLO	Able to complete the task over 80% (4-5)	Able to complete the task 50-80% (2-3)	Able to complete the task below 50% (0-1)	Marks
1. Realization of experiment	3	Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed.	Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed.	Does not execute due to syntax errors, runtime errors, user prompts are misleading or non- existent. No testing has been completed.	
2. Conducting experiment	2	Able to make changes and answered all questions.	Partially able to make changes and few incorrect answers.	Unable to make changes and answer all questions.	
3. Computer use	4	Document submission timely.	Document submission late.	Document submission not done.	
4. Teamwork	4	Actively engages and cooperates with other group member(s) in effective manner.	Cooperates with other group member(s) in a reasonable manner but conduct can be improved.	Distracts or discourages other group members from conducting the experiment	
5. Laboratory safety and disciplinary rules	2	Code comments are added and does help the reader to understand the code.	Code comments are added and does not help the reader to understand the code.	Code comments are not added.	
6. Data collection	2	Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap.	Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables.	Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy.	
7. Data analysis	3	Solution is efficient, easy to understand, and maintain.	A logical solution that is easy to follow but it is not the most efficient.	A difficult and inefficient solution.	
Total (out of 35):					