Department of Computer and Software Engineering- ITU
CE101L: Object Oriented Programming Lab

Course Instructor: Nadir Abbas	Dated: 04/02/2025
Teaching Assistant: Zainab, Nahal, Bilal	Semester: Spring 2025
Session: 2024-2028	Batch: BSCE24

Lab 3. Getting Familiar With JSON File Handling With Wtructures And Nested Structures

Name	Roll number	Obtained Marks/35

Checked on: ______________________________

Signature: ______________________________

Objective

The objective of this lab is to help students understand and apply concept of structs and nested structs.

Equipment and Component

Component Description	Value	Quantity
Computer	Available in lab	1

Conduct of Lab

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

Theory and Background

Nested Structures

A nested structure is a structure that contains another structure as a member. This helps in organizing related data efficiently. For example, a Student structure can have a Course structure inside it, allowing better representation of real-world relationships.

JSON File Handling

JSON (JavaScript Object Notation) is a lightweight format for storing and exchanging data. In C++, we use the nlohmann/json library for JSON serialization (converting a struct to JSON) and deserialization (loading JSON into a struct).

Steps:

1. Convert a struct to JSON and save it to a file.
2. Read the JSON file and populate the struct.

Example snippet:

```
json j;
j["name"] = student.name;
j["course"]["courseName"] = student.course.courseName;
ofstream file("student.json");
file << j.dump(4);

 ifstream file("student.json");
 json j; file >> j;
 Student student;
 student.name = j["name"];
 student.age = j["age"];
 student.course.courseName = j["course"]["courseName"];
```

Tasks

Task 1: Accept the assignment posted in Google Classroom and after accepting clone the repository to your computer for this ensure you have logged into github app with your account.

Task 2: Solve the given problems written after task instructions, write code through IDE like CLion

Task 3: Ensure your code/solution is in the cloned folder.

Task 4: Commit and Push the changes through the Github App

Write code in functions, in files **functions.cpp**, **functions.h** and **main.cpp** after completing each part, verify through running code using **"make run"** on Cygwin.

Question 1.

Structure Definitions:

1. Structure: Course

Members:

- string courseName (Course Name)
- int courseCode (Course Code)

Constructors:

- **Default Constructor:** Initializes courseName as "" and courseCode as 0.
- **Parameterized Constructor:** Takes values for courseName and courseCode.

2. Structure: Student

Members:

- string name (Student's Name)
- int age (Student's Age)
- float gpa (Student's Grade Point Average)
- Course course (Nested structure: Course)

Constructors:

- **Default Constructor:** Initializes name as "", age as 0, gpa as 0.0, and assigns a default Course object.
- **Parameterized Constructor:** Takes values for name, age, gpa, and a Course object.

Question 2.

void assignCourseToStudent(Student& s, const Course& c)

- Assigns a Course to a Student.

Question 3.

Student* compareStudents(const Student& s1, const Student& s2)

- Compares two students based on their GPA and returns the student pointer with highest GPA.

Question 4.

void saveToFile(const Student& studentObj, const string& filename)

- Saves the Student structure object to a JSON file.

Question 5.

void loadFromFile(Student* studentObj, const string& filename)

- Reads from a JSON file and populates the Student structure object.

Question 6.

void updateGPA(Student& s, float newGPA)

- Updates the GPA of a student.

Example JSON File Structure:

Sample JSON file (student_data.json)

```
{
  "name": "John Doe",
  "age": 20,
  "gpa": 3.8,
  "course": {
    "courseName": "Object Oriented Programming",
    "courseCode": 101
  }
}
```

Additional Notes:

- Use #include <iostream> and #include <fstream> for file operations.
- Use #include “json.hpp” for JSON handling.
- Use namespace std; and using json = nlohmann::json; in your implementation.

Assessment Rubric for Lab

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	1	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	1	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	2	Document submission timely.	Document submission late.	Document submission not done.	
4. Teamwork	3	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	3	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	3	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	4	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):					