Department of Computer and Software Engineering – ITU

CE101T: Object Oriented Programming

Course Instructor: Nadir Abbas	**Dated: 02/02/2025**
Teaching Assistant: Zainab, Nahal, Bilal	**Semester: Spring 2025**
Session: 2024-2028	**Batch: BSCE2024**

Assignment 2. Implementation of Structures and Nested Structures

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

Note: You have to use code from lab 2 for this assignment.

Q1: Define Structures

1. **Project:**
    - **Members:**
        - **projectId (integer)**
        - **projectName (string)**
        - **status (bool)**
    - **Constructors:**
        - **Default Constructor: Initializes projectId to 0, projectName to an empty string, and status to false.**
        - **Parameterized Constructor:**
          Project (int id, string name, bool status)
          Accepts values for all fields and initializes them using an initialization list.
2. **Performance:**
    - **Members:**
        - **rating (float)**
        - **feedback (string)**
    - **Constructors:**
        - **Default Constructor: Initializes rating to 0 and feedback to an empty string.**
        - **Parameterized Constructor:**
          Performance (float rating, string feedback)
          Accepts values for all fields.
3. **Employee:**
    - **Members:**
        - **empId (integer)**
        - **name (string)**
        - **department (string)**
        - **projects (*Project array of project struct)**
        - **numProjects (int)**
        - **performance (Performance)**
    - **Constructors:**
        - **Default Constructor: Initializes empId to 0 name, department to an empty string projects to nullptr, numProjects to 0 performance initialized using the Performance structure's default constructor.**
        - **Parameterized Constructor:**
          Employee (int id, string name, string department, Performance performance)
          Accepts id, name, department, and performance as input parameters.
            - **Initializes the respective fields using an initialization list.**

- **Sets projects to nullptr and numProjects to 0 as no projects are assigned initially.**

4. **Company:**

   - **Members:**
     - **name (string)**
     - **numEmployees (int)**
     - **employees (Employee* array of employees)**
   - **Constructors:**
     - **Default Constructor: Initializes name to empty string numEmployees to 0 employees to nullptr**
     - **Parameterized Constructor:**
       Company (string name)
       Accepts the name of the company as an input parameter.
       - **Initializes name using an initialization list.**
       - **Sets numEmployees to 0 and employees to nullptr by default.**

Q2: List All Projects of an Employee

Implement the following function:

Project* listEmployeeProjects(Employee& employee)

Task:

- **Accept an Employee object by constant reference.**
- **Return all the active project list or nullptr if no active projects.**

Q3: Find the Top-Performing Employee

Implement the following function:

Employee findTopPerformer(const Company& company)

Task:

- **Accept a Company object.**
- **Iterate through all employees and return the one with the highest performance rating. if more than one return first employee with highest ratting**

Q4: Count Active Projects in the Company

Implement the following function:

int countActiveProjects(const Company& company)

Task:

- **Accept a Company object.**
- **Count and return the total number of active projects across all employees.**

Q5: Remove an Employee from a Company

Implement the following function:

void removeEmployee(Company& company, int empId)

Task:

- **Accept a Company object and an employee ID (empId).**
- **Remove the employee with the given ID from the company's employees array.**
- **Update the numEmployees field.**

Q6: Search an employee in the company

Implement the following function:

bool searchEmployee(Company& company, int empId)

Task:

- **Accept a Company object and an employee ID (empId).**
- **search the employee with the id and return 0 if not present and 1 if present**

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):					