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

Course Instructor: Nadir Abbas	Dated: 11/02/2025
Teaching Assistant: Zainab, Nahal, Bilal	Semester: Spring 2025
Session: 2024-2028	Batch: BSCE24

Lab 4. Introduction to Classes and Constructors, Static Variables And Inline Functions

Name	Roll number	Obtained Marks/35

Checked on: ______________________________

Signature: ______________________________

Objective

The objective of this lab is to help students understand and apply concepts of classes, constructors, destructors, inline functions, static variables, and copy constructors..

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

1. Class Introduction

A **class** is a blueprint for creating objects, encapsulating data members and functions to define object behavior and attributes.

2. Constructor and Destructor

A **constructor** initializes objects when created, while a **destructor** releases resources when an object is destroyed.

```
class Example {
public:
   Example() { cout << "Constructor called"; }  // Constructor
   ~Example() { cout << "Destructor called"; }  // Destructor
};
```

3. Static Variables

A **static variable** is shared among all class instances and retains its value across function calls.

```
class Counter {
   static int count;
public:
   Counter() { count++; }
   static int getCount() { return count; }
};
int Counter::count = 0;
```

6. Copy Constructor

A **copy constructor** is a special constructor used to create a new object as a copy of an existing object.

```
class CopyExample {
   int x;
public:
   CopyExample(int val) { x = val; }
   CopyExample(const CopyExample &obj) { x = obj.x; }
};
```

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

Define a class **Book** with the following specifications:

Members:

- string **title** (Book Title)
- string **author** (Book Author)
- int **pages** (Number of Pages)

Constructors:

- **Default Constructor**: Initializes title as "", author as "", and pages as 0.
- **Parameterized Constructor**: Takes values for title, author, and pages.
- **Copy Constructor**: Initializes a new Book object using an existing Book object.
- **Destructor**.

Question 2

Implement the following inline member function for the Book class:

- **bool isThickBook():** Returns true if the number of pages is greater than 500, otherwise false.

Question 3

Implement setter and getter functions for each data member of the Book class:

- void setTitle(string t)
- string getTitle()
- void setAuthor(string a)
- string getAuthor()
- void setPages(int p)
- int getPages()

Question 4

- Define and initialize the static variable **bookCount** inside the class.
- Modify the constructors and destructor to update bookCount accordingly.
- Implement the function **static int getBookCount()** that returns the current value of bookCount.

Question 5

- Create a function **Book duplicateBook(const Book& original)** that takes a Book object as a parameter and creates a copy using the copy constructor and returns it.

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):					