**National University of Computer and Emerging Sciences **

**Laboratory Manual # 10**

**Object Oriented Programming**

| Course Instructor | Mr. Uzair Naqvi |
| --- | --- |
| Lab Instructors | Seemab Ayub , Muhammad Hashir |
| Section | BCS-2B |
| Date | 16-April-2024 |
| Semester | Spring-24 |

**Instructions for lab submission:**

You have to submit source code (.cpp) files along with a word document. In the word document you have to give the heading of each exercise/question, then paste your source code and output snippet. Save your word document in the following format: roll number-lab no-section i.e. 22l-0008-lab10-BCS2B.

**Objectives:**

In this lab students will practice:

**● Inheritance**

● Implement a base class and derived classes in C++

● Understand the working of base class and derived class constructors/destructors. ● Understand the working and purpose of private and protected attributes of a base class. ● Use base class member functions in a derived class.

**1) Exercise: Marks: 10** Consider the following hierarchy as it exists in a university:

a) There are two types of persons in the university i.e. Student and Faculty b) Every Person has some basic information that is common to all persons i.e. the first\_name and last\_name stored as private attributes and age which is a protected attribute.

c) A student can in turn be either an Undergraduate or a Graduate student, every student has a cgpa.

d) An undergraduate student has a fyp\_name as his private attribute.

e) A graduate student has a thesis topic as his private attribute.

f) A faculty member has private attributes about the number of courses he is currently teaching, i.e. his course\_count and a three digit telephone extension number.

Implement the entire hierarchy of the classes i.e. define all the classes along with their attributes and their inheritance, add getters and setters for all attributes in all the classes that you have defined. Also write the main() function to show the basic functionality.

**2) Exercise: Marks: 10**

Add appropriate constructors and destructors to all the classes created in Exercise 1. For example the constructor for the Person class should take three inputs (for **first\_name**, **last\_name** and **age**). The student constructor should take four inputs, three for its parent class (i.e. Person) and one float value to be assigned to the **cgpa** attribute.

This is accomplished in the following manner:

Person (char\* fname, char\* lname, int age)

{

...

cout << ”Person() called”;

}

Student (char\* fname,char\* lname,int age,float cgpa): Person(fname,lname,age) {

...

cout << ”Student() called”;

}

This syntax can be generalized to any parent and child constructor accordingly. Following this syntax, define and implement constructors and destructors for all the classes. Also, Notice that you have to add a print statement in every constructor which announces that the constructor has been called.

Also add a print statement to every destructor which announces that the destructor has been called. For example, the destructor for Person should look like:

~Person()

{

cout << ”~Person() called”;

}

**3) Exercise: Marks: 10**

Your task is to create a simple banking system using C++ inheritance. The system should handle different types of bank accounts and their transactions.

Base Class: Account

● Create a base class called Account to represent generic properties of bank accounts.

● Include attributes such as accountId, accountType, balance, and interestRate. ● Implement methods to set and get the attributes, as well as deposit and withdraw funds.

Derived Classes: Different Types of Accounts

● Derive classes representing different types of bank accounts from the Account base class. Examples include SavingsAccount, CheckingAccount, and

FixedDepositAccount.

● Each derived class should have additional attributes specific to its type (e.g., minimumBalance for savings accounts, overdraftLimit for checking accounts, termLength for fixed deposit accounts).

● Implement methods to set and get the additional attributes, and override methods like withdraw() to account for specific account rules.

Additional Features:

● Implement constructors and destructors for all classes to ensure proper initialization and cleanup.

● Demonstrate the use of inheritance by creating objects of different classes and accessing their attributes and methods.

● Test your implementation by creating scenarios involving various account types and transactions.

Your task is to design the class hierarchy, implement the required classes, and create a C++ program that demonstrates the functionality of the banking system.