Student ID: ____________

AIR UNIVERSITY

Air University
Mid Semester Examinations: Spring 2025

Subjective Part
(To be solved on Answer Books only)

Subject: Object-Oriented Programming
Class: BSCYS
Section(s): A
Course Code: CS-112

Time Allowed: 180 Minutes
Max Marks: 100
FM's Name: Sheikh Qaisar Ayyub
FM's Signature:

INSTRUCTIONS

- Attempt responses on the answer book only.
- Nothing is to be written on the question paper.
- Rough work or writing on question paper will be considered as use of unfair means.
- Tables / calculators are allowed / not allowed.

Q. No. 1 (CLO 1) — 20 Marks

Question (a) — 10

onsider the following specific types of students in a university system:

- **UndergraduateStudent**
- **PostgraduateStudent**
- **PhDStudent**

Each of these student types has some **common attributes and behaviors** (e.g., name, ID, enrollInCourse()) but also **distinct features** (e.g., thesisTitle for PhDStudent, internshipDetails for UndergraduateStudent, and researchArea for PostgraduateStudent).

Based on this scenario, answer the following:

1. Define the concept of **generalization** in object-oriented design and identify common attributes in above scenario.
2. Draw a class diagram with common attributes and connectivity.

```
#include <iostream.h>
#include <conio.h>
class Vehicle {
  char make[20];
  char model[20];
  void startEngine() {
    cout << "Starting engine of " << make << " " << model << endl;
  }
};
class Car : public Vehicle {
  int doors;
  void display() {
    cout << "Car with " << doors << " doors" << endl;
    startEngine(); // Error 1
  }
};
class Bike : public Vehicle {
  int gear;
  void display() {
    cout << "Bike with " << gear << " gears" << endl;
    startEngine(); // Error 2
  }
};
void main() {
  clrscr();
  Car c;
  c.display();
  Bike b;
  b.display();
  getch(); }
```

Question (b) — 10

Identify Compile time errors in above code

20

Q. No. 2 (CLO 2)

Question (a)

You are part of a development team building a modular Java application for a Library Management System. The system is divided into multiple packages:

- library.users – contains classes like Member, Librarian
- library.books – contains classes like Book, Catalog
- library.utils – contains helper classes like DateUtils, Logger

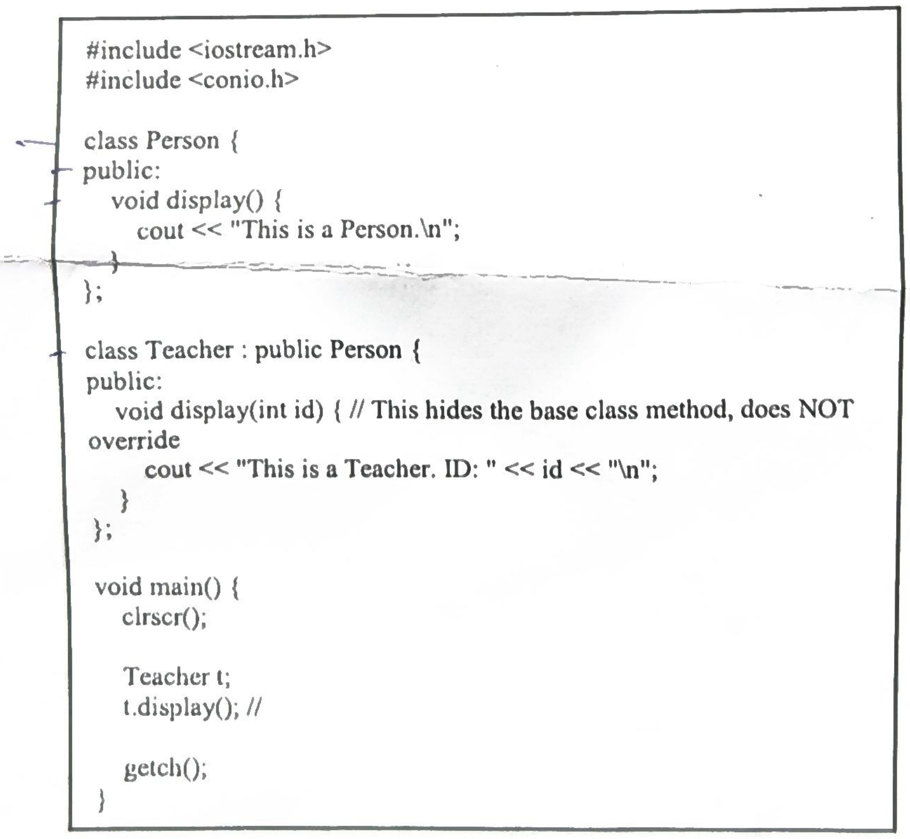
While working on the project, a junior developer wrote the following:

- Made some class attributes private, some public
- Declared the Logger class with no access specifier
- Tried to access the Logger class from the library.books.Book class
- Exposed a method from Member to Librarian via package-level access

10

Based on this scenario, answer the following:

1. What would happen when the library.books.Book class tries to use the Logger class from library.utils, and why?
2. Explain the difference between public, private, and default access in Java with reference to classes and class members.



```
#include <iostream.h>
#include <conio.h>

class Person {
public:
   void display() {
      cout << "This is a Person.\n";
   }
};

class Teacher : public Person {
public:
   void display(int id) { // This hides the base class method, does NOT
override
      cout << "This is a Teacher. ID: " << id << "\n";
   }
};

void main() {
   clrscr();

   Teacher t;
   t.display(); //

   getch();
}
```

10

Question(b)

Compile the above code and point out which function is invoked with t.display();

Q. No. 3 (CLO 03)

20

Question:

You are developing a console-based academic system using **Turbo C++ 3.2** for a small college. The system includes various types of users such as **Students**, **Teachers**, and **Admins**. All these user types share common behaviors like login() and logout(), but each performs different actions like:

- **Student**: viewGrades()
- **Teacher**: enterMarks()
- **Admin**: manageSystem()

20

To ensure **consistent design and function declarations**, you decide to use **an abstract base class** to define the interface for all user types.

Based on this scenario, answer the following:

1. What is an **abstract class** in C++, and how is it declared in **Turbo C++ 3.2**?
2. How would you define a base class User with **pure virtual functions** to enforce that all derived classes implement login() and logout()?
3. Why is it not possible to create an object of an abstract class?
4. If a class inherits from User but does not implement all pure virtual functions, what will happen in Turbo C++ 3.2 and why?

Q. No. 4 (CLO 03)

10

Question:

You are building a Java application for an **Online Payment System**. The system must support different types of payment methods such as:

- **Credit Card**
- **PayPal**
- **Bank Transfer**

Each payment method must provide implementations for the following actions:

- authenticate()
- processPayment()
- generateReceipt()

To ensure a **consistent structure** across all payment methods, the development team decides to use an **interface** in Java.

Based on this scenario, answer the following:

1. What is an **interface** in Java, and how does it differ from an abstract class?
2. How does using an interface improve the flexibility and maintainability of the application?

10

Q. No. 5 (CLO 02)

30

Question:

You are building a simple console-based student utility program in Turbo C++ 3.2 that can display student information in multiple formats:

- Display only the student's roll number
- Display the roll number and name
- Display roll number, name, and marks

To achieve this, you decide to use function overloading. Here's the sample code:

```
#include <iostream.h>
#include <conio.h>
class Student {
public:
    void display(int rollNo) {
        cout << "Roll Number: " << rollNo << endl;
    }
    void display(int rollNo, char name[]) {
        cout << "Roll Number: " << rollNo << ", Name: "
<< name << endl;
    }
    void display(int rollNo, char name[], int marks) {
        cout << "Roll Number: " << rollNo << ", Name: "
<< name << ", Marks: " << marks << endl;
    }
};

void main() {
    clrscr();
    Student s;
    s.display(101);
    s.display(102, "Ali");
    s.display(103, "Sara", 95);
    getch();
}
```

20

Based on this scenario and code, answer the following theoretical questions:

1. What is function overloading, and how is it demonstrated in the above example?
2. How does Turbo C++ 3.2 determine which display() function to call?
3. Can function overloading be done by changing only the return type in Turbo C++? Explain your answer.
4. What are the advantages of using function overloading in a student information system?
5. What will happen if two functions are written like this in the class?

```
void display(int rollNo);
int display(int rollNo); // valid or not? Explain.
```

Question: What will be the output in blew program, is there any logical error ?

```
public class MarksCalculator {

  public void calculateMarks() {
    int total = 0;  // Local variable (method-level scope)
      {
      int math = 85;  // Block-level variable
      total = total + math;
    }
    {
      int science = 90;  // Another block-level variable
      total = total + science;
    }
    System.out.println("Total Marks: " + total);
    System.out.println("Math Marks: " + math);     //
ERROR
    System.out.println("Science Marks: " + science); //
  }

  public static void main(String[] args) {
    MarksCalculator m = new MarksCalculator();
    m.calculateMarks();
  }
}
```

88

175