University of Ruhuna- Faculty of Technology

Bachelor of Engineering Technology Honours Degree
Level 2 (Semester I) Examination, June/July 2026
Academic year 2024/2025

Course Unit: ENT2142 Object Oriented Programming (Theory) **Duration: 2 hours**

- This is a closed book examination.
- This examination consists of four (04) questions on **six (06)** pages. Answer all questions.
- Use C++ programming language for answering questions.

1)

a) Define high-level programming languages with an example.

(2 mark)

b) Briefly describe the difference between interpreted languages and compiled languages based on their execution process.

(4 mark)

c) Define the terms ***class*** and ***object*** in Object-Oriented Programming. Support each definition with a suitable real-world example that clearly illustrates the relationship between the above two.

(6 mark)

d) Explain the purpose of the ***protected access specifier*** in object-oriented programming and describe how it differs from the ***private access specifier***.

(2 mark)

e) A school requires a software system to manage its staff. Write C++ code based on the specifications below.

i) Define a class named *Teacher* with private attributes: *teacherID*, *name*, and *subject*. Include a public method *displayDetails()* to display the teacher's details.

ii) Define a class named *Student* with private attributes: *studentID*, *name*, and *marks* to store marks of three subjects. Include a public method *displayDetails()* to display the student's details along with the marks.

iii) Define a class named *Exam*. Then define a method named *confirmExam()* outside the class. When called, it should print "Exam confirmed.".

(11mark)

2)

a) Explain the difference between a ***"default constructor"*** and a ***"parameterized constructor"***.

(4 mark)

b) A smart home application requires a class to represent a *TemperatureSensor*. Follow the steps below to implement it in C++ programming.

i) Declare a class named *TemperatureSensor* with the following private attributes: *sensorID*, *location*, and *currentTemp*. Also, declare an attribute *sensorCount* to track the total number of sensor objects created.

(3 mark)

ii) Implement constructors with the following specifications,

a. A constructor that initializes all data members to appropriate default values and increments *sensorCount*.

b. A constructor that accepts values for all three (03) attributes and increments *sensorCount*.

(5 mark)

iii) Implement a destructor that decrements *sensorCount* and displays "Sensor destroyed" and the current active sensor count. (2 mark)

iv) Write a static member function *getCount()* to return the value of *sensorCount*. Show how to call this function without creating an object. (3 mark)

v) Create one object using the default constructor and another object using the parameterized constructor. Then call the *getCount()* function. (3 mark)

c) For the given code **Q2.Code 01**

i. Declare the class *Display* as a friend class of the *Student* class. (3 mark)

ii. Write a function inside the Display class that shows the marks of a student.

(2 mark)

```
class Student {
private:
    int marks;

public:
    Student(int m) {
        marks = m;    }
};

class Display {
public:
};
```

Q2.Code 01

3)

a) Explain the concept of **polymorphism** in object-oriented programming and name the two (02) main types of polymorphism.

(3 marks)

b) Describe the purpose of virtual functions in achieving runtime polymorphism,

(2 marks)

c) The incomplete program in **Q3.Code 01** represents a *Box* class that uses operator overloading to add the volumes of two Box objects together. Complete the operator+ function and write down the expected output.

(4 marks)

```
##include <iostream>
using namespace std;

class Box {
public:
    int volume;

    Box(int v) {
        volume = v;
 }
        Box operator+(Box obj) {
    }
};

int main() {
    Box b1(10), b2(5);
    Box b3 = b1 + b2;
    cout << "Total Volume: " << b3.volume << endl;
    return 0;
}
```

Q3.Code 01

d) Answer the questions below based on the program in **Q3.Code 02.**

i. Explain the reason for the *sound()* method being declared as virtual in the base class.

ii. What will be the output of the program if the virtual keyword is removed from the base class function? Write the output clearly.

(4 mark)

```
#include <iostream>
using namespace std;

class Animal {
public:
    virtual void sound() {
        cout << "Animal makes a sound" << endl;
    }
};

class Dog : public Animal {
public:
    void sound() {
        cout << "Dog barks" << endl;
    }
};

class Cat : public Animal {
public:
    void sound() {
        cout << "Cat meows" << endl;
    }

};
int main() {
    Animal* a1 = new Dog();
    Animal* a2 = new Cat();
    a1->sound();
    a2->sound();
    return 0;
}
```

Q3.Code 02

e) Explain the functionality of multiple inheritance using a block diagram.

(3 mark)

f) Answer the questions below based on the program in **Q3.Code 03.**

i. Identify the type of inheritance used in the above given code and clearly mention the base and derived classes. (3 mark)

ii. List all the members (data members and member functions) that are accessible from the member functions of the class *Car*. (4 mark)

iii. Suppose that the class *Car* was derived privately from both Vehicle and Insurance. Name the functions that could be accessed through objects of class *Car*. (2 mark)

```
#include <iostream>
using namespace std;

class Vehicle {
public:
    int engineCapacity;
    void start() {
        cout << "Vehicle started" << endl;
    }
    void showDetails() {
        cout << "Engine Capacity: " << engineCapacity << endl;
    }
};

class Insurance {
public:
    float coverageAmount;
    void showPolicyInfo() {
        cout << "Coverage Amount: " << coverageAmount << endl;
    }
};
class Car : public Vehicle, public Insurance {
public:
    int carID;
    string ownerName;
    void displayCarInfo() {
        cout << "Car ID: " << carID << endl;
        cout << "Owner: " << ownerName << endl;
    }
    void scheduleService() {
        cout << "Service scheduled." << endl;
    }
};
```

Q3.Code 03

4)

a) Explain the concept of encapsulation relevant to object-oriented programming and list down two (02) advantages of encapsulation.

(4 marks)

b) Clearly explain the difference between **Encapsulation** and **Abstraction** in C++.

(2 marks)

c) List down two (02) disadvantages of using a traditional file-based system compared to a Relational Database Management System (DBMS).

(2 marks)

d) List down three (02) components that are commonly found in a DBMS architecture.

(3 marks)

e) Define the concept of composite attributes in an ***Entity-Relationship (ER)*** model and provide an example.

(2 marks)

f) When implementing a database management system for a hospital appointment management system, the Patients and Doctors tables are included with the following columns (fields),

Patients – PATIENT_ID, PATIENT_NAME, DATE_OF_BIRTH, CONTACT_NO, EMAIL, ASSIGNED_DOCTOR

Doctors – DOCTOR_ID, DOCTOR_NAME, SPECIALIZATION, CONTACT_NO, DEPARTMENT

i. Identify the **primary keys** of these two (02) tables and justify your selection.

ii. Provide a SQL query to create the *Doctors* table with appropriate data types.

iii. Provide a SQL query to retrieve *PATIENT_NAME* and *CONTACT_NO* of all patients from the *Patients* table who are assigned to the department "*Cardiology*".

iv. Provide a SQL query to update the email address of a patient named "*Nimal*" to *nimal@hospital.lk* in the *Patients* table.

v. Determine the foreign key for the *Patients* table and justify your selection.

(12 marks)

.....End of the paper.....