UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 2 Examination in Engineering: March 2026

Module Number: EE2201 Module Name: Computer Programming II Part-B

C-18 (Repeat)

[Part-A + Part-B = 2 hours]

Instructions for candidates

- Answer all questions
- Write the index number very clearly in each answer sheet.
- For Q1 answer in the seperate sheet Provided.
- For Q2 and Q3 answer in the booklet provided.

Q1. Complete the following C++ programs by filling in the blanks. Choose the correct term for each blank from the word list given for the corresponding question. (Each correct blank is 0.5 marks.)

a) The code defines a class with a parameterized constructor, a private member variable, and a method. It then allocates an object dynamically in the `main` function.

```
->          b          Device      getDeviceInfo      new
private     string
```

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

class Device {
    (1)
.....:
    string brand;

public:
    // Parameterized constructor
    (2)
.....(string b) {
        brand = (3)
.....;
    }

    // Method returning a formatted string
```

```
    ..... getDeviceInfo() {
        return "Device Brand: " + brand;
    }
};

int main() {
    // Dynamic object creation
    Device* myDevice = ..... Device("Tablet");

    // Calling the method via pointer
    string info = myDevice..... .....();

    cout << info << endl;

    delete myDevice;
    return 0;
}
```

b) The following code demonstrates C++ inheritance, polymorphic object creation using pointers, and method overriding.

```
->          new          override      protected      public
Vehicle::   virtual
```

```
class Vehicle {
// Accessible to derived classes but not externally
.....:
    int topSpeed;

public:
    // Allows derived classes to provide their own implementation
    ..... void StartEngine() {
        cout << "Vehicle engine starting..." << endl;
    }
};

// Derived class inheriting publicly from the base class
class Car : ..... Vehicle {
public:
    // Modifying the inherited method
    void StartEngine() ..... {
        // Calling the original method from the parent class
        .....StartEngine();
        topSpeed = 120;
        cout << "Car is ready to drive." << endl;
```

```
    }
};

int main() {
    // Polymorphic object creation
    Vehicle* myCar = ..... Car();

    // Executing the method
    myCar ..... StartEngine();

    delete myCar;
    return 0;
}
```

c) The following C++ program iterates through an array of integers to find and print the maximum value.

=	>	cout	for	if
numbers[0]	numbers[i]	size		

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

int main() {
    int numbers[] = {12, 45, 7, 89, 23};
    int size = 5;

    // Assume the first element is the maximum to start
    int maxVal = .....;

    // Iterate through the rest of the array
    ..... (int i = 1; i < .....; i++) {
        // Check if the current element is greater than maxVal
        ..... (numbers[i] ..... maxVal) {
            // Update the maximum value
            maxVal ..... .....;
        }
    }

    ..... << "The maximum value is: " << maxVal << endl;
    return 0;
}
```

d) The following C++ program demonstrates passing variables to functions using pointers and references, as well as accessing their values.

&	&num1	*	*p	*ptr
cout	num2	return		

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

// Function taking a pointer parameter
void updateValue(int* ptr) {
    // Dereference the pointer to assign a new value
    (23) = 20;
}

// Function taking a reference parameter
void doubleValue(int (24) ref) {
    ref = ref * 2;
}

int main() {
    int num1 = 5;
    int num2 = 10;

    // Pass the memory address of num1
    updateValue((25));

    // Pass num2 by reference directly
    doubleValue((26));

    // Declare a pointer and assign it the address of num1
    int (27) p = &num1;

    // Dereference the pointer to print its value
    (28) << "num1 is now: " << (29) << endl;

    cout << "num2 is now: " << num2 << endl;

    (30) 0;
}
```

Q2. Design a C++ class named `Thermostat` to manage a room's temperature. Follow the requirements below to demonstrate encapsulation and basic class structure:

a) Write the class definition for `Thermostat` with the following requirements:

- Create a private integer field named `currentTemp`.
- Create a public integer field named `maxTemp`.
- Implement a public constructor that takes one integer parameter to initialize `maxTemp`, and automatically sets `currentTemp` to a default of 20.

[1.5 marks]

b) Implement getter and setter methods to securely encapsulate the temperature with the following logic:

- Create a public method `int getTemp()` that returns the current temperature.
- Create a public method `void setTemp(int t)`. In this method, write logic ensuring that if `t` is less than 10, the temperature is set to 10. If `t` is greater than `maxTemp`, it is set to `maxTemp`. Otherwise, assign the exact value of `t`.

[2 marks]

c) Write a method to simulate turning up the heat based on the following rules:

- Create a public method named `void increaseTemp(int amount)`.
- Inside the method, calculate the new temperature by adding `amount` to the current temperature.
- Use your `setTemp` method to assign this new value so it automatically follows your safety limits.

[2 marks]

d) Write a C++ `main()` function to demonstrate the usage of your class by performing the following actions in order:

- Instantiate a `Thermostat` object with a `maxTemp` of 30.
- Intentionally call `setTemp(35)` to test the upper boundary logic.
- Call the `increaseTemp(5)` method.
- Print the final temperature to the console using `cout`.

[2 marks]

Q3. Object-oriented inheritance allows us to build upon existing code. Design a simplified C++ class hierarchy for an employee payroll system demonstrating inheritance and polymorphism:

a) Write a base class named `Employee` with the following requirements:

- Create a `protected` string field named `name`.
- Implement a public constructor that takes a string parameter to initialize the `name`.

- Declare a public `virtual` method named `getPay()` that returns a `double` value of `500.0`.
- Include a `virtual` destructor.

[1.5 marks]

b) Write a class named `Manager` that inherits publicly from `Employee` with the following requirements:

- Create a private `double` field named `bonus`.
- Implement a public constructor taking a string for the name and a double for the bonus.
- Pass the name parameter to the base class constructor, and initialize the `bonus` field.

[2 marks]

c) Implement the specific payroll logic for the manager:

- `override` the `getPay()` method inherited from the base class.
- Inside the method, calculate and return the total pay by calling the base class's `getPay()` method and adding the `bonus` to it.

[2 marks]

d) Write a C++ `main()` function to demonstrate late binding using pointers:

- Create an array or `vector` of `Employee*` pointers.
- Dynamically allocate (`new`) one `Employee` and one `Manager` object, and add them to your collection.
- Iterate through the collection using a loop, calling `getPay()` on each pointer and printing the result to the console.
- Properly `delete` the dynamically allocated objects to prevent memory leaks.

[2 marks]