

Object-oriented Programming (CS-1004)

Sessional-I Exam

Date: 24/2/2025

Total Time: 1hr

Course Instructor(s)

Total Marks: 30

Basit Ali, Minhal, Sumaiya, Nida, Sobia, Abeeha, Rafia, Atiya, Abeer, Bakhtawar.

Total Questions: 02

Roll No

Section

Student Signature

Do not write below this

Attempt all questions.

CLO 1: Discuss knowledge of underlying concepts of object-oriented paradigm like abstraction, encapsulation, polymorphism, inheritance etc.

[Marks 05, 10 min]

Q1 (a): Answer the following question .

- i. Why can't copy constructors accept objects by value?

Copy constructors can't accept objects by value because pass-by-value requires a copy, which calls the copy constructor; this process repeats as a new copy is created each time, leading to infinite recursion.

- ii. Why is it not possible to call a constructor explicitly for an already created object in C++? What mechanisms ensure the integrity of object construction?

In C++, constructors cannot be explicitly called on an already created object because they are meant for one-time initialization and reinitializing an object this way would corrupt its state. Mechanisms like automatic constructor invocation during object creation and restriction on direct reinitialization ensure object integrity.

- iii. Is it permissible to invoke a destructor directly in C++? If so, what potential issues might arise, and under what conditions should it be done, if ever?

Yes, a destructor can be invoked directly in C++. Typically, destructors are called using `delete` for dynamically allocated objects to ensure proper deallocation and prevent memory leaks.

- iv. In C++, when multiple objects call functions of the same class, how does the compiler distinguish between them and associate function calls with the correct object?

In C++, the compiler distinguishes between multiple objects calling functions of the same class using the hidden `this` pointer, which holds the address of the calling object and ensures function calls operate on the correct instance.

- v. In object-oriented programming, are member functions considered a part of each object instance of a class? Provide a detailed explanation with justification.

No, member functions are not part of each object instance; they belong to the class itself and exist only once in memory, shared by all instances. Each object has its own copy of non-static data.

members, but all instances use the same function code, with the `this` pointer ensuring the function operates on the correct object's data.

CLO 2: Identify real world problems in terms of objects rather than procedure.

Q1 (b): Predict the output or Identify Error of the following programs.

[Marks 1.25*8, 20 min]

<p>(i)</p> <pre>#include <iostream> using namespace std; class ABC { int a, b, c; public: ABC(int x = 5, int y=6, int z=7) :a(x), b(y), c(z) { } void display() { cout << "a: " << a << ", b: " << b << ", c: " << c << endl; } }; main() { ABC obj(10); obj.display(); } output: 10,6,7</pre>	<p>(ii)</p> <pre>class Employee{ public: Employee() { cout << "Employee Constructor"; } ~Employee() { cout << "Employee Destructor"; } }; class Company{ private: Employee obj; public: Company(Employee obj) { this->obj = obj; cout << "Company Constructor"; } ~Company() { cout << "Company Destructor"; } }; int main() { Employee eobj; { Company cobj(eobj); } }</pre>
<p>(iii)</p> <p>If I change the definition of above function with</p> <pre>ABC(int x = 5, int y=6, int z) :a(x), b(y), c(z) { } Then what will be the output?</pre> <p>Output: Error, solution can assign default values from left most</p>	<p>output</p> <p>Employee Constructor Employee Constructor Company Constructor Employee Destructor Company Destructor Employee Destructor Employee Destructor</p>
<p>(iv)</p> <pre>class Test { int value; public: Test(int value) : this->value(value) {} void display() { cout << "Value:" << value; } }; int main() { Test obj(10); obj.display(); return 0; }</pre> <p>Output: Error expected identifier before 'this'</p>	<p>(v)</p> <pre>int main() { const int num = 50; const int *const ptr = &num; *ptr = 60; int anotherNum = 70; ptr = &anotherNum; cout << "Value of num: " << *ptr; return 0; }</pre> <p>Output: assignment of read-only variable 'ptr'</p>

<p>(vi) .</p> <pre>int main() { const int ptr_1 = 10; int ptr_2 = 20; int *ptr_3 = &ptr_2; const int* ptr_4 = ptr_3; ptr_3 = &ptr_2; *ptr_4 = ptr_1; cout << *ptr_3; cout << *ptr_4; return 0; }</pre> <p>output:assignment of read-only location '* ptr_4'</p>	<p>(vii)</p> <p>In part (ii), if I change the Employee object in the Company class to a pointer, what modifications are needed for the code to function correctly?</p> <p>Output: Company(Employee &obj){ obj = obj; cout <<"Company Constructor";}</p> <hr/> <p>(viii)class Constants { int x; const int y; public: Example(int x, int y) : x(x) { this -> y = y;} };</p> <pre>int main() { Constants c (1,2); }</pre> <p>Output: y should also be initialize by initializer list</p>
---	--

CLO 4: Illustrate Object-Oriented design artifacts and their mapping to Object-Oriented Programming using C++.

[Marks 15, 30 min]

Q2: Design a system for managing User Accounts, Messages and WhatsApp Groups using object-oriented principles.

- Create a class for Users with attributes such as name, about, and phone number. Make a parameterized constructor, getters and setters. **(2 points)**
- A Message class stores the sender_name, text, and timestamp. Make a default constructor, getters and setters methods. **(2 points)**
- The class WhatsAppGroup should have data members such as group_info, group_name, creation_date, an array for admin_list (5 max) , and an array for member_list (max 20).A WhatsAppGroup contains only the last 10 messages (space allocated through DMA).If a group is deleted, its messages should also be deleted. A user can exist independently without a group. **(2 points)**
- Make a parameterized constructor, copy constructor and destructor. **(3 points)**
- The WhatsAppGroup class should also include the following functions.
 - add_admin(User *u) which can add any already existing member to adminlist. A group can have many users and one or more admins. At any given time there must be at least one admin per group. Admin must be an existing member of the group. **(2 points)**
 - remove_admin(User *u) which can remove an admin from adminlist. At any given time there must be at least one admin per group. Admin must be an existing member of the group. **(2 points)**
 - send_message(Message m) which can send any new message to the list **(2 points)**