National University of Computer and Emerging Sciences
Karachi Campus

Object-oriented Programming (CS-1004) Sessional-I Exam

Date: 24/2/2025

Course Instructor(s)

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

Total Time:	1hr
Total Marks:	**30**
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]**

(i)

```
#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
Or error "cout << "a: " << a << ", b: " << b << , c: " << c << endl;" use of comma

(ii)

```
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);}}
```

output

```
Employee Constructor
Employee Constructor
Company Constructor
Employee Destructor
Company Destructor
Employee Destructor
Employee Destructor
```

(iii)

If I change the definition of above function with

```
    ABC(int x = 5,int y=6,int z)
:a(x),b(y),c(z){
}
```

Then what will be the output?

Outpu:Error,solution can assign default values from left most

(iv)

```
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;
}
```

Output:Error expected identifier before 'this'

(v)

```
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;
}
```

Output: assignment of read-only variable 'ptr'

(vi).
```
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;
}output:assignment of read-only
location '* ptr_4'
```

(vii)
```
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?
Output: Company(Employee &obj){
                obj = obj;
                cout <<"Company
Constructor";}
or
Company(Employee &obj){
                this->obj = &obj;
                cout <<"Company
Constructor";}
or
       Company(Employee *obj){
                this->obj = obj;
                cout <<"Company
Constructor";}
int main(){
      Employee eobj;
      {
      Company cobj(&eobj);}}
```

(viii)
```
class Constants {
    int x; const int y;
public:
    Example(int x, int y ) : x(x) {
        this -> y = y;} };
int main() {
   Constants c (1,2); }
Output: y should also be initialize by
initializer list, constructor name
should be class name.
```

***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)**

```
// Sp2025_Mid_1_Sol.cpp : This file contains the 'main' function. Program
execution begins and ends there.
//

#include <iostream>
using namespace std;

class User {
      string name;
      string about;
      string phone_num;

public:
    //Parameterized Constructor - [REQUIRED]
    User(string name = "", string about = "", string phone_num = "") {
        this->name = name;
        this->about = about;
        this->phone_num = phone_num;
    }

    //Copy Constructor that I'm using later in WhatsAppGroup - Not Required
    User(const User& obj) {
        this->name = obj.name;
        this->about = obj.about;
        this->phone_num = obj.phone_num;
    }
    // Setters - [REQUIRED]
    void setName(const string& newName) {
        name = newName;
    }

    void setAbout(const string& newAbout) {
        about = newAbout;
    }

    void setPhoneNum(const string& newPhoneNum) {
        phone_num = newPhoneNum;
    }

    // Getters - [REQUIRED]
    string getName() const {
        return name;
    }

    string getAbout() const {
        return about;
    }

    string getPhoneNum() const {
        return phone_num;
    }
};

class Message {
    string sender;
    string text;
    string timestamp;
```

```
public:
    //default constructor for Message - [REQUIRED]
    Message() {
        sender = "";
        text = "";
        timestamp = "";
    }
    //Copy Constructor that I'm using later in WhatsAppGroup - not required
    Message(const Message& obj) {
        sender = obj.sender;
        text = obj.text;
        timestamp = obj.timestamp;
    }
    string getsender() const {
        return sender;
    }
    string gettext() const {
        return text;
    }
    string gettimestamp() const {
        return timestamp;
    }
    void setSender(const string& sender) {
        this->sender = sender;
    }
    void setText(const string& text) {
        this->text = text;
    }
    void setTimestamp(const string& timestamp) {
        this->timestamp = timestamp;
    }
};

class WhatsAppGroup {
    string group_info;
    string group_name;
    string creation_date;
    User* admin_list[5];
    User* member_list[20];
    Message* message_list;
    int message_list_size;
    int message_count=0;
public:
    //not required by question, but making it cuz might need it to compile the
code
    WhatsAppGroup() {
        group_info = "";
        group_name = "";
        creation_date = "";
        for (int i = 0; i < 5; i++)
            admin_list[i] = nullptr;
        for (int i = 0; i < 20; i++)
            member_list[i] = nullptr;
        message_list_size = 0;
        message_list = nullptr;
    }

    //Parameterized Constructor  - [REQUIRED]
    WhatsAppGroup(string gi, string gn, string cd, User* al[5], User* ml[20],
Message* mel, int mls) {
        group_info = gi;
        group_name = gn;
        creation_date = cd;
        for (int i = 0; i < 5; i++)
            admin_list[i] = al[i];
        for (int i = 0; i < 20; i++)
            member_list[i] = ml[i];
        message_list_size = mls;

        //DMA for messages - [ABSOLUTELY REQUIRED]
        message_list = new Message [message_list_size];
        for (int i = 0; i < message_list_size; i++) {
            message_list[i] = mel[i];
        }

    }

    //Copy Constructor - [REQUIRED]
    WhatsAppGroup(const WhatsAppGroup &obj) {
        this->group_info = obj.group_info;
        this->group_name = obj.group_name;
        this->creation_date = obj.creation_date;

        //not doing DMA because it aggregation, so we're copying pointers.
        for (int i = 0; i < 5; i++)
            admin_list[i] = obj.admin_list[i];
        for (int i = 0; i < 20; i++)
            member_list[i] = obj.member_list[i];

        this->message_list_size = obj.message_list_size;

        //DMA the list - [ABSOLUTELY REQUIRED]
        message_list = new Message [message_list_size];
        for (int i = 0; i < message_list_size; i++) {
            message_list[i] = obj.message_list[i];
        }
    }

    //Add Admin - [REQUIRED]
    void add_admin(User* u) {
        for (int i = 0; i < 20; i++) {
            if (member_list[i] && member_list[i]->getName() == u->getName()) {
//checking for null to be safe
                int x = 0;
                while (x < 5) {
                    if (!admin_list[x]) { // if it is null, that means there is
still space for another admin
                        cout << "Admin added" << endl;
                        admin_list[x] = u;
                        break;
                    }
                    x++;
                }
                if (x >=5) {
                    cout << "list full" << endl;
                }
                return;
            }
        }
        cout << "User not in list\n";
    }

    //Remove Admin - [REQUIRED]
    void remove_admin(User* u) {
        int count = 0;
        for (int i = 0; i < 5; i++) {
            if (admin_list[i])
                count++;
        }
        cout << count << endl;
        if (count > 1) { // ensuring that we don't remove the last admin
[REQUIRED]
            for (int i = 0; i < 5; i++) {
                if (admin_list[i] && admin_list[i]->getName() == u->getName())
{
                    admin_list[i] = nullptr;
                    cout << "Admin Removed\n";
                    return;
                }

            }
            cout << "User not in admin list\n" << endl;
        }
        else {
            cout << "Cannot Remove Last Admin\n";
        }
    }

    //Send Message - [REQUIRED]
    void send_message(Message m) {
     if (message_count<9)
     {
         message_list[message_count++]=m;
         cout<<"message added "<<endl;
     }
     else
     {int i;
         for( i=0; i<9; i++)
         {
             message_list[i]=message_list[i+1];
         }
         message_list[i]=m;
         cout<<"message added "<<endl;
     }
    }


    //Destructor for WhatsAppGroup - [REQUIRED - CHECK FOR MESSAGE DELETION]
    ~WhatsAppGroup() {
        delete message_list; // deleting the DMA'd message pointers
    }
};

int main()
{
    User u1("A"), u2("B"), u3("C");
    User* mem_arr[20] = { &u1, &u2, &u3, nullptr, nullptr, nullptr, nullptr,
nullptr, nullptr, nullptr,
                        nullptr, nullptr, nullptr, nullptr, nullptr, nullptr,
nullptr, nullptr, nullptr, nullptr };
    User* adm_arr[5] = { &u1, &u2, &u3, nullptr , nullptr };
    Message* mList;
    mList = new Message [10];

    WhatsAppGroup w_g = WhatsAppGroup("info", "name", "24/2/2025", adm_arr,
mem_arr, mList, 10);
    w_g.add_admin(&u1);
    w_g.add_admin(&u2);
    w_g.add_admin(&u3);
    w_g.remove_admin(&u1);
    w_g.add_admin(&u3);
    w_g.remove_admin(&u1);
    w_g.remove_admin(&u2);
    w_g.remove_admin(&u2);
    w_g.remove_admin(&u3);
    w_g.remove_admin(&u3);//removing the last admin
    Message m1;
    w_g.send_message(m1);
}
```