National University of Computer and Emerging Sciences

Karachi Campus

Object Oriented Programming (CS1004)

Final Exam

Date: May 20th 2024

Total Time (Hrs): 3

Total Marks: 100

Total Questions: 06

Course Instructor(s)

Mr. Basit, Mr. Minhal, Ms.Sumaiya, Ms.Sobia, Ms.Abeeha, Ms.Bakhtawer, Ms.Abeer, Ms.Atiya, Ms.Rafia

_____ _____ _____

Roll No Section Student Signature

Do not write below this line

Attempt all the questions.

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

Q1: Write short answers (2-3 lines) for the following questions: **[10 minutes, 1*5=5 marks]**

a) What is the difference between compile-time error and runtime error? Which types of errors can be dealt with using exceptions?
b) Which principle of object-oriented programming suits in the below scenarios?
    1. A bank vault with multiple layers of security protecting valuable assets.
    2. A car with different driving modes for different road conditions.
c) Write an advantage of using overloading instead of generics in C++?
d) Can a friend function of Base class access the data members of a derived class?
e) If class C inherits class B and class B inherits class A, then Class C object can be upcasted to object of either class A or B?

CLO #3: Illustrate Object-Oriented design artifacts and their mapping to Object-Oriented Programming using C++

Q2: Predict the output of the following code snippets. If there are any errors, provide a proper justification. Exclude errors related to libraries. **[25 minutes, 2*5=10 marks]**

A.
```
class Test
{ public: Test() { cout <<"Hello from Test() "; }
} a;
int main()
{ cout <<"Main Started "; }
```

B.
```
class Test {
public:
   Test() { cout << "Constructing an object of Test " << endl; }
  ~Test() { cout << "Destructing an object of Test "  << endl; }
};
int main() {
  try {
    Test t1;
```

```
        throw 10;
      } catch(int i) { cout << "Caught " << i << endl; } }
```

C.
```
template <typename T>
void fun(const T&x)
{   static int count = 0;
    cout << "x = " << x << " count = " << count;
    ++count;
    return;
}
int main()
{
    fun<int> (1); cout << endl;
    fun<int>(1); cout << endl;
    fun<double>(1.1); cout << endl;
}
```

D.
```
int main() {
try {
     cout<<"Throwing Block"<<endl;
    throw 'x';
}
catch (int e) {
cout <<"Catching Int" << e << endl; }
catch (...) {
cout <<"Catching Default" << endl; }
catch (char e) {
cout <<"Catching Char" << e << endl; }
 return 0;}
```

E.
```
char foo(int i) {
    if (i % 3 != 0)
        return 't';
    throw 'o';
    return 'p';
}
int main() {
    char c = 'c';
    try {
        for (int i = 1; ; i++) { c = foo(i); cout << "c" << endl;}
         }
    catch (char ex) {
        cout << c << endl; cout << ex << endl; } }
```

CLO #3: Illustrate Object-Oriented design artifacts and their mapping to Object-Oriented Programming using C++

Q3: Complete the C++ code for an Object-Oriented Programming (OOP) scenario.

[25 minutes, 5*2=10 marks]

A. For the given class, you are required to create a specialized template that manages computations specifically when both arrays are characters with a size of 10. Overload the function so that it returns a string containing all elements of arr1 followed by all elements of arr2.

```
template <class T, int size>
class QuestionTemplate {
    T arr1[size];
    T arr2[size];
public:
    QuestionTemplate() {
        //assume numbers only for now
        for (int i = 0; i < size; i++){
            arr1[i] = i;
            arr2[size - i - 1] = i;
        }
    }
    T* add() {
        T* arr = new T[size];
        for (int i = 0; i < size; i++)
            arr[i] = arr1[i] + arr2[i];
        return arr;
    }
};

int main() {
    QuestionTemplate <int, 10> qt;
    int* res = qt.add();
    for (int i = 0; i < 10; i++)
        cout << res[i] << endl;
    QuestionTemplate <char, 10> ct;
    cout << ct.add();
}
```

B. Complete the given code below in such a way that when we run this code, a similar kind of output is stored in the file.

outfile.txt:
ID = 1, Name = 0001
ID = 2, Name = 0002

```
class Test {
    int ID;
    string name;
    static int genID;
public:
};

int Test::genID = 0;

int main() {
    Test t1, t2;
    t1 + "outfile.txt";
    t2 + "outfile.txt";
}
```

CLO #4: Design and assess small and medium scale C++ / C# programs using object-oriented programming principles

Q4: Sui Southern Gas Company (SSGC) is one of the leading natural gas utility companies in Pakistan, responsible for distributing natural gas to various regions. As part of its operations, SSGC maintains a fleet of vehicles for meter reading, pipeline inspection, and maintenance tasks. To enhance operational efficiency and ensure proper management of the fleet, SSGC has decided to implement a Fleet Management System (FMS). This system will track vehicle locations, monitor fuel consumption, analyze driver performance, and manage maintenance logs. **[40 minutes, 4+4+5+4+4+4=25 marks]**

A. Create **TelemetryData Class** simulated telemetry data for vehicles , including attributes : The amount of fuel consumed by the vehicle, Fuel Consumption (liters per hour),The current speed of the vehicle Speed (kilometers per hour).The status of the vehicle's engine Engine Status(true if running, false otherwise).

B. Design a **Vehicle Class** to store the vehicle details, including attributes Vehicle ID ("V1234") ,Model ("Toyota Corolla 2022").Fuel Type ("Petrol" or "Diesel"), Current Location (“Karachi”) and Instance of Telemetry Data class
Implement the functions as described below
   - Virtual displayDetails(): Displays detailed information about the vehicle, including its ID, model, fuel type, current location, and telemetry data .
   - updateLocation(const string& newLocation): Updates the current location of the vehicle with the provided new location.
   - getVehicleID() const: Retrieves the unique identifier of the vehicle.

C. Create a friend class of vehicle as **VehicleManager Class**: Responsible for tracking the location of vehicles using trackVehicleLocation(Vehicle& vehicle): Tracks the location of a vehicle and displays the current location of the vehicle with its ID.
   i. Create function WriteToFile(vehicle * vehiclesArray): writes the details of all vehicles provided to a file named "vehicle_info ".
   b. Create function ReadFromFile(): Read those vehicle from file whose location is **Karachi**.

D. **MeterReadingVehicle Class** Subclass of Vehicle, representing a vehicle used specifically for meter reading include attributes Reading Capacity(int), Overrides displayDetails() to display reading capacity.

E. **PipelineInspectionVehicle Class**:Subclass of Vehicle, representing a vehicle used for pipeline inspection include attribute InspectionRange(int), Overrides displayDetails() to dispay inspection range.

F. **MaintenanceVehicle Class**: Subclass of Vehicle Represents a vehicle used for maintenance purposes, include attribute Equipment Capacity (int): Capacity of the vehicle to carry maintenance equipment, measured in units. Overrides displayDetails() to include equipment capacity.

CLO #4: Design and assess small and medium scale C++ / C# programs using object-oriented programming principles

Q5: Imagine you have been asked to develop a system for managing the Jewelry shop using Object-Oriented Programming (OOP) concepts. The system should handle various types of jewelry items, customer transactions, and inventory management. Consider the following detailed requirements for the system:

[45 minutes, 6*4+6=30 marks]

1. JewelryItem Class (Abstract):

- Create an abstract class called JewelryItem.
- Attributes: itemCode (string), itemName (string), weightInGrams (double), purity (int).
- Methods:

- virtual void displayDetails(): Display the details of the jewelry item including item code, item name, weight, and purity.
- virtual double calculatePrice(): Calculate and return the price of the jewelry item based on weight and purity.

2. GoldJewelry Class (Derived from JewelryItem):

- Inherits from the JewelryItem class.
- Additional Attribute: goldKarat (int).
- Methods:
  - void setGoldKarat(int karat): Set the gold karat value for the item.
  - Override calculatePrice() to calculate the price based on weight, purity, and gold karat.

3. DiamondJewelry Class (Derived from JewelryItem):

- Inherits from the JewelryItem class.
- Additional Attribute: numDiamonds (int), diamondCarat (double).
- Methods:
  - void addDiamonds(int num, double carat): Add diamonds to the jewelry item with the specified carat weight.
  - Override calculatePrice() to calculate the price based on weight, purity, and diamond carat weight.

4. Customer Class:

- Create a class called Customer to represent customers visiting the jewelry store.
- Attributes: customerID (string), name (string), purchasedItems (dynamic pointer to JewelryItem objects).
- Methods:
  - void purchaseItem(JewelryItem* item): Add a purchased item to the customer's list of purchased items.
  - double calculateTotalPurchasePrice(): Calculate the total price of all purchased items.

5. StoreInventory Class:

- Create a class called StoreInventory to manage the store's inventory of jewelry items.
- Attributes: inventory (dynamic pointer to JewelryItem objects).
- Methods:
  - void addItemToInventory(JewelryItem* item): Add a jewelry item to the store's inventory.
  - void displayInventory(): Display the details of all items in the inventory.

6. Operator Overloading:

- Overload the comparison operators (<, >, ==) for JewelryItem objects based on their price (calculated using calculatePrice()).

7. Draw UML/Class Diagram of the requirements given above.

CLO #5: Synthesize programs using Generic Programming and exception handling

Q6: Imagine you are developing a shopping cart system for an e-commerce website. The shopping cart should be able to hold up to 10 unique items of different product types that are book, electronics & clothing. Create and implement a template class named UniqueCart to manage the shopping cart's functionalities for various product types. **[30 minutes, 20 marks]**

A. **Add:** Implement a method to add items to the cart (void add(const T& item)), where T represents three different product types that are books, electronics & clothing.
B. **Remove:** Implement a method to remove items from the cart (void remove(const T& item)).
C. **Contains:** Implement a method to check if a specific item exists in the cart (bool contains(const T& item)). Your implementation should also handle the exceptions for the following scenarios:
    a. Throw an exception "Duplicate Item Exception" when trying to add a duplicate item to the cart.
    b. Throw an exception "Item Not Found Exception" when trying to remove an item that does not exist in the cart.
    c. Throw an exception "Out of bound" when the cart capacity (10 items) is exceeded.