National University of Computer and Emerging Sciences
Peshawar Campus

Object Oriented Programming

Date: 16th December, 2024

Course Instructor(s)

Qasim Jan, Sara Rehmat, Usman Wajid

Final- Exam

Total Time (Hrs):	3
Total Marks:	100
Total Questions:	8

Roll No _____ Section _____ Student Signature _____

Do not write below this line

Attempt all the questions.

CLO 1: Demonstrate the basic concepts of OOP.

Q1. Write the short answers of the following: Answer should not be more than 3 lines. **[10 Marks]**

a. Can **static members** be accessed without creating an **object** of the class? Justify.
b. How does **method/function overloading** differ from **operator overloading** in C++?
c. What is an **abstract class**? Can we create objects of the abstract class?
d. What is a **function template**? Specify when its use is most appropriate.

CLO 2: Apply OOP concepts to computing problems for related programs.

Q2. Complete the following code and write only the completed code, mentioning the line number: **[20 Marks]**

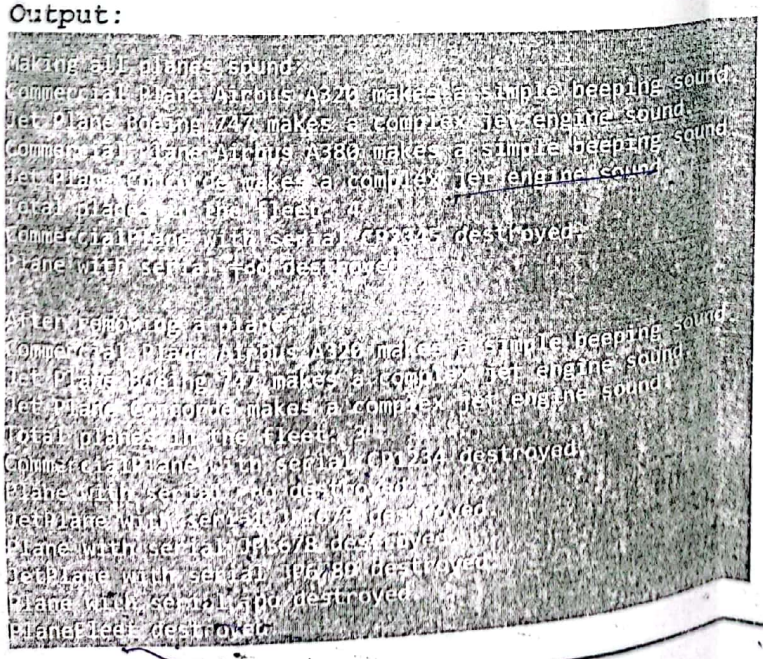
Note: Do not write the entire code. Mention the line number and function name, and then write the corresponding code.

```
class Plane {
public:
    char* serialNumber;
    char* name;
    int yearManufactured;

    // Constructor
    Plane(const char* sn, const char* n, int year) {
      // Line 1 complete this code
    }

    virtual void sound() = 0;

    virtual ~Plane() {
    // Line 2 complete this code
    }
};

class CommercialPlane : public Plane {
public:
    CommercialPlane(const char* sn, const char* n, int year {
    // Line 3 complete this code
    }
```

```
    void sound() override {
    // Line 4 complete this code
    }

    CommercialPlane() {
    // Line 5 complete this code
    }
};

class JetPlane : public Plane {
public:
    JetPlane(const char* sn, const char* n, int year){
    // Line 6 complete this code
    }

    void sound() override {
    // Line 7 complete this code
    }

    ~JetPlane() {
    // Line 8 complete this code
    }
};

class PlaneFleet {
private:
    Plane** planes;
    int planeCount;
    int capacity;

public:
    PlaneFleet() {
        planeCount = 0;
        capacity = 5;  // Initial capacity for the fleet
        planes = new Plane*[capacity];  // Dynamic array to hold Plane pointers
    }

    void addPlane(Plane* plane) {
        // Resize the array if the fleet is full
        if (planeCount >= capacity) {
            // Line 9 complete the code
        }

        planes[planeCount++] = plane;
    }

    void removePlane(int index) {
        // Line 10 complete the code
    }

    void makeAllPlanesSound() {
        // Line 11 complete the code
    }

    void displayPlaneCount() {
        cout << "Total planes in the fleet: " << planeCount << endl;
```

```
    }
    ~PlaneFleet() {
    // Line 12 complete this code
    }
};

int main() {
    // Create the plane fleet
    PlaneFleet fleet;

    // Add planes to the fleet
    fleet.addPlane(new CommercialPlane("CP1234", "Airbus A320", 2020));
    fleet.addPlane(new JetPlane("JP5678", "Boeing 747", 2021));
    fleet.addPlane(new CommercialPlane("CP2345", "Airbus A380", 2022));
    fleet.addPlane(new JetPlane("JP6789", "Concorde", 2023));

    // Make all planes sound
    cout << "\nMaking all planes sound:" << endl;
    fleet.makeAllPlanesSound();

    // Display the total number of planes
    fleet.displayPlaneCount();

    // Remove a plane and show the updated fleet
    fleet.removePlane(2);  // Remove the third plane

    cout << "\nAfter removing a plane:" << endl;
    fleet.makeAllPlanesSound();
    fleet.displayPlaneCount();

    // Cleanup handled manually, destructors will be called
    return 0;
}
```

Output:



Q3: Rewrite the following code and replace inheritance with composition. **[8 Marks]**

```
class Person {
public:
    string name;
    int age;
Person(string n, int a) : name(n), age(a) {}
    void display() { cout << "Name: " << name << ", Age: " << age << endl; }
};
class Student : public Person {
public:
    string schoolName;

Student(string n, int a, string school) : Person(n, a), schoolName(school) {}

    void displayStudentInfo() {
        display();
        cout << "School: " << schoolName << endl;
    }
};
```

CLO 3: Model an algorithmic solution for a given problem using OOP.

Q4: A weakness of C++ is that it does not automatically check array indexes to see whether they are in bounds. (This makes array operations faster but less safe.) We can use a class to create a safe array that checks the index of all array accesses. **[17 Marks]**

Write a class called **SafeArray** that uses an int array of fixed size (call it **LIMIT**) as its only data member. There will be two member functions. The first, **putel()**, takes an index number and an int value as arguments and inserts the int value into the array at the index. The second, **getel()**, takes an index number as an argument and returns the int value of the element with that index.

- Overload the subscription operator for SafeArray class so that array elements can be accessed using the subscription operator.
- Create objects of **SafeArray** class in the main method and use methods as well as subscription operators to set and get the values of array elements.
- Change the entire class to **template class** so that the array elements can be of any type.
- Rewrite main function that creates objects of class **SafeArray** for at least two different types and calls its methods.

Q5: Create a fractionType class that represents fractions. Each fraction has two integers: numerator and denominator. **[15 Marks]**

- Define constructor(s).
- Define a method that reduces the fraction object of which it is a member to lowest terms. It finds the greatest common divisor (gcd) of the fraction's numerator and denominator and uses this gcd to divide both numbers.
- Overload basic arithmetic operators for addition and multiplication of fractions.
- Overload equality operator for comparison of fractions.
- Overload stream insertion and stream extraction operators for input and output of fractions.

CLO 1. Understand principles of object oriented paradigm.

Q6: Identify the errors, write only the corrected code, mention the line number. [10 Marks]

a.

```
1. class Alpha {
2. int y;
3. protected:
4. int x; // Protected member

5. public:
6. Alpha() : x(10), y(20) {}
7. void showAlpha() {
8. cout << "Alpha Class: x = " << x
   << endl;}};

9. class Beta : public Alpha {
10. public:
11. void showBeta() {
12. cout << "Beta Class: x = " << x <<
    endl;
13. cout << "y = " << y << endl;
14. }};

15. class Gamma : private Alpha {
16. public:
17. void showGamma() {
18. cout << "Gamma Class: x = " << x
    << endl;
19. cout << "y = " << y << endl;
20. }};

21. class Epsilon : public Gamma {
22. protected:
23. void showEpsilon() {
24. cout << "Epsilon Class: x = " << x
    << endl;
25. }
26. };

27. class Zeta {
28. public:
29. void showZeta(const Alpha&
    alphaObj) {
30. cout << "Zeta Class accessing
    Alpha: x = " << alphaObj.x << ", y
    = " << alphaObj.y << endl;
31. }
32. };

33. int main() {
34. Beta b;
35. showAlpha();
36. showBeta();

37. Gamma g;
38. g.showAlpha();
39. g.showGamma();

40. Epsilon e;
41. showAlpha();
42. showEpsilon();

43. Zeta z;
44. z.showZeta(a);

45. return 0;
46. }
```

CLO 4: Apply good programming practices.

Q7: Identify the errors, write only the corrected code, mention the line number. Also apply good programming practices while rewriting the code. [10 Marks]

```
1. class Shape {
2. public:
3. double area() const = 0;
4. };

5. class Circle : public Shape {
6. private:
7. double radius;

8. public:
```

```
9.  Circle(double r) : radius(r) {}

10. double area()  {
        return 2 * M_PI * radius;
11. }
12. };
13. class Square : public Shape {
14. private:
15. double side;

16. public:
17. Square(double s) {}


18. double area()  {
19. }
20. };
```

```
21. int main() {
22. Shape shapes[2];

23. shapes[0] = new Circle(3.0);
24. shapes[1] = new Square(4.0);

25. for (int i = 0; i <= 2; ++i) {
26. cout << "Area: " << shapes[i]->area()
    << endl;

27. }

28. return 0;
29. }
```

Q8: Dry run the following code and write the output. [10 Marks]

```
int *intArrayPtr;
int *temp;
intArrayPtr = new int[5];
*intArrayPtr = 7;
temp = intArrayPtr;
for (int i = 1; i < 5; i++)
{
intArrayPtr++;
*intArrayPtr = *(intArrayPtr - 1) + 2 * i;
}
```

```
intArrayPtr = temp;
for (int i = 0; i < 5; i++)
{
cout << *intArrayPtr << " ";
intArrayPtr++;
}
cout << endl;
```