**Roll No…………….. Total No. of Pages:……**

**ST-2 (SET-I)**

**4th SEMESTER 2022-23**

**22CS006- Object Oriented Programming**

**Time allowed: 90 Minutes Max. Marks: 40**

**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1. What is the correct syntax to declare a class in C++?
   1. **class MyClass;**
   2. class MyClass()
   3. class() MyClass
   4. MyClass class
2. In C++, an object is:
   1. **An instance of a class**
   2. A method in a class
   3. A variable inside a class
   4. A function returning a value
3. Which access specifier allows members to be accessed by any code in the program?
   1. **public**
   2. private
   3. protected
   4. global
4. What is the purpose of inheritance in C++?
   1. To create new classes
   2. To hide data members
   3. **To reuse and extend the functionality of existing classes**
   4. To create private members
5. What is the other name used for functions inside a class?
   1. Member variables
   2. **Member functions**
   3. Class functions
   4. Class variables
6. A constructor in C++:
   1. **Is automatically called when an object is created**
   2. Is used to destroy objects
   3. Can be inherited from a base class
   4. Is used to allocate memory for an object
7. How many constructors can a class have in C++?
   1. Only one default constructor
   2. Only one parameterized constructor
   3. Multiple default constructors
   4. **Multiple constructors with different parameters**
8. What is the purpose of a destructor in C++?
   1. To create objects
   2. **To deallocate memory and perform cleanup before an object is destroyed**
   3. To initialize class members
   4. To overload operators
9. Which operator is overloaded for a cout object?
   1. >>
   2. **<<**
   3. <
   4. >
10. What is the access specifier used by default for class members in C++?
    1. public
    2. **private**
    3. protected
    4. static

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

1. Which of the following feature of OOPs is not used in the following C++ code?

class A{

int i;

public:

void print() {cout << "hello" << i;}

}

class B : public A{

int j;

public:

void assign (int a ) {k = a;}

}

a) Abstraction

b) Encapsulation

c) Inheritance

d) **Polymorphism**

1. What is the output of the following code?

#include <iostream>

using namespace std;

class Base {

public:

Base() { cout << "Base constructor" << endl; }

~Base() { cout << "Base destructor" << endl; }

};

class Derived : public Base {

public:

Derived() { cout << "Derived constructor" << endl; }

~Derived() { cout << "Derived destructor" << endl; }

};

int main() {

Base\* ptr = new Derived();

delete ptr;

return 0;

}

**a) Base constructor, Derived constructor, Derived destructor, Base destructor**

b) Derived constructor, Base constructor, Base destructor, Derived destructor

c) Base constructor, Derived constructor, Base destructor

d) Derived constructor, Base destructor

1. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class complex

{

int i;

int j;

public:

complex(){}

complex(int a, int b)

{

i = a;

j = b;

}

complex operator+(complex c)

{

complex temp;

temp.i = this->i + c.i;

temp.j = this->j + c.j;

return temp;

}

void show(){

cout<<"Complex Number: "<<i<<" + i"<<j<<endl;

}

};

int main(int argc, char const \*argv[])

{

complex c1(1,2);

complex c2(3,4);

complex c3 = c1 + c2;

c3.show();

return 0;

}

**a) Complex Number: 4 + i6**

b) Complex Number: 2 + i2

c) Error

d) Segmentation fault

1. Using friend operator function, following perfect set of operators may not be overloaded.

**a) = , ( ) , [ ] , ->**

b) <<, = = , [ ] , >>

c) ?, = , ( ) , ++

d) None of these

1. Which of the following statements are not true about destructor?

1. It is invoked when object goes out of the scope

2. Like constructor, it can also have parameters

3. It can be virtual

4. It can be declared in private section

5. It bears same name as that of the class and precedes Lambda sign.

a) Only 2, 3, 5

b) Only 2, 3, 4

**c) Only 2, 4, 5**

d) Only 3, 4, 5

**SECTION-C(Coding Question) (2x5 marks=5 marks)**

Q16) Alice is one baker who is making pastries. She wants precise measurements of fractions. She wants to add two measurement of two pastries.

Create a class 'Fraction' with attributes 'numerator' and 'denominator'. Implement functions to add two fractions and display the result in simplified form to help Alice.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | frac1(3, 5)  frac2(2, 7) | frac1(3, 7)  frac2(4, 7) | frac1(5, 8)  frac2(2, 4) |
| **Output** | Fraction: 31/35 | Fraction: 1/1 | Fraction: 9/8 |

Solution :

**#include <iostream>**

**class Fraction {**

**private:**

**int numerator;**

**int denominator;**

**public:**

**// Constructor**

**Fraction(int n, int d) : numerator(n), denominator(d) {}**

**// Function to add fractions**

**void add(const Fraction& other) {**

**numerator = numerator \* other.denominator + other.numerator \* denominator;**

**denominator \*= other.denominator;**

**simplify();**

**}**

**// Function to find greatest common divisor (GCD)**

**int gcd(int a, int b) {**

**if (b == 0)**

**return a;**

**return gcd(b, a % b);**

**}**

**// Function to simplify the fraction**

**void simplify() {**

**int common = gcd(numerator, denominator);**

**numerator /= common;**

**denominator /= common;**

**}**

**// Function to display the fraction**

**void display() {**

**std::cout << "Fraction: " << numerator << "/" << denominator << std::endl;**

**}**

**};**

**int main() {**

**Fraction frac1(5, 8);**

**Fraction frac2(2, 4);**

**frac1.add(frac2);**

**frac1.display();**

**return 0;**

**}**

Q17) You work as a software engineer for a company specializing in CAD (Computer-Aided Design) software.

Your team has been assigned a task to create a C++ program that handles 3D shapes.

The program should support two types of 3D shapes: spheres and cones.

The main goal is to calculate and display the volumes of these shapes using method overriding.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | Sphere(radius=5)  Cone(radius=3, height=7) | Sphere(radius=5)  Cone(radius=3, height=6) | Sphere(radius=4)  Cone(radius=6, height=9) |
| **Output** | Volume of Sphere: 523.333  Volume of Cone: 65.94 | Volume of Sphere: 523.333  Volume of Cone: 56.52 | Volume of Sphere: 267.947  Volume of Cone: 339.12 |

Solution :

**#include <iostream> // Include the standard input/output stream library**

**#include <cmath> // Include the math library for mathematical functions**

**#define M\_PI 3.14 // Define the constant value of pi**

**// Base class for 3D shapes**

**class Shape3D {**

**public:**

**// Virtual function to calculate the volume of the shape (to be overridden by derived classes)**

**virtual double calculateVolume() const {**

**return 0.0; // Default implementation returns 0 for the volume**

**}**

**};**

**// Sphere class, derived from Shape3D**

**class Sphere : public Shape3D {**

**private:**

**double radius; // The radius of the sphere**

**public:**

**// Constructor to initialize the sphere with a given radius**

**Sphere(double r) : radius(r) {}**

**// Overridden function to calculate the volume of the sphere**

**double calculateVolume() const override {**

**return (4.0 / 3.0) \* M\_PI \* pow(radius, 3); // Volume formula for a sphere**

**}**

**};**

**// Cone class, derived from Shape3D**

**class Cone : public Shape3D {**

**private:**

**double radius; // The radius of the cone**

**double height; // The height of the cone**

**public:**

**// Constructor to initialize the cone with given radius and height**

**Cone(double r, double h) : radius(r), height(h) {}**

**// Overridden function to calculate the volume of the cone**

**double calculateVolume() const override {**

**return (1.0 / 3.0) \* M\_PI \* pow(radius, 2) \* height; // Volume formula for a cone**

**}**

**};**

**int main() {**

**// Creating instances of Sphere and Cone objects**

**Shape3D\* shape1 = new Sphere(5);**

**Shape3D\* shape2 = new Cone(3, 6);**

**// Calculating and displaying the volume of each shape using their respective functions**

**std::cout << "Volume of Sphere: " << shape1->calculateVolume() << std::endl;**

**std::cout << "Volume of Cone: " << shape2->calculateVolume() << std::endl;**

**// Cleaning up the allocated memory for the objects**

**delete shape1;**

**delete shape2;**

**return 0;**

**}**

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q18) You work at a space research center where precise timing is crucial for coordinating space missions.

The scientists and engineers often need to perform time calculations to synchronize mission events accurately.

To help them with their tasks, you decide to create a class named 'Time' that will handle time-related addition.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | t1(h=2, m=30, s=45)  t2(h=1, m=45, s=15) | t1(h=1, m=45, s=45)  t2(h=2, m=15, s=15) | t1(h=5, m=15, s=60)  t2(h=2, m=45, s=5) |
| **Output** | Time: 4h 16m 0s | Time: 4h 1m 0s | Time: 8h 1m 5s |

Solution :

**#include <iostream>**

**class Time {**

**private:**

**int hours;**

**int minutes;**

**int seconds;**

**public:**

**// Constructor**

**Time(int h, int m, int s) : hours(h), minutes(m), seconds(s) {}**

**// Function to add time**

**void addTime(const Time& other) {**

**seconds += other.seconds;**

**minutes += seconds / 60;**

**seconds %= 60;**

**minutes += other.minutes;**

**hours += minutes / 60;**

**minutes %= 60;**

**hours += other.hours;**

**}**

**// Function to display time**

**void displayTime() {**

**std::cout << "Time: " << hours << "h " << minutes << "m " << seconds << "s\n";**

**}**

**};**

**int main() {**

**Time t1(5, 15, 60);**

**Time t2(2, 45, 5);**

**t1.addTime(t2);**

**t1.displayTime();**

**return 0;**

**}**