1.)

STEP – 1: Creating a base class ‘Shape’ with a pure virtual functional ‘area’ which must be overridden by derived classes.

#include <iostream>

#include <cmath>

#include <vector>

class Shape {

public:

Shape() {}

virtual ~Shape() {} // Virtual destructor

virtual double area() const = 0; // Pure virtual function for calculating area

bool operator==(const Shape& other) const { // Overload == operator

return this->area() == other.area(); }

};

Step 2: Derived Class Rectangle - We define the Rectangle class derived from Shape, with constructors and an overridden area () function.

class Rectangle : public Shape {

private:

double length;

double width;

public:

Rectangle(double l, double w) : length(l), width(w) {}

~Rectangle() {}

double area() const override {

return length \* width; }

};

Step 3: Derived Class Circle - We define the Circle class derived from Shape, with constructors and an overridden area () function.

class Circle : public Shape {

private:

double radius;

public:

Circle(double r) : radius(r) {}

~Circle() {}

double area() const override {

return M\_PI \* radius \* radius; }

};

Step 4: Friend Function total Area - We define a friend function total Area that takes a vector of Shape pointers and calculates the total area of all shapes.

double totalArea(const std::vector<Shape\*>& shapes) {

double total = 0.0;

for (const Shape\* shape : shapes) {

total += shape->area(); }

return total;

}

Step 5: Optional Template Class Point - We define a template class Point to represent a point in 2D space.

template <typename T>

class Point {

public:

T x, y;

Point(T xCoord, T yCoord) : x(xCoord), y(yCoord) {}

void display() const {

std::cout << "(" << x << ", " << y << ")\n"; }

};

Step – 6 : Main Function for Testing - Finally, we create objects of Rectangle and Circle, test their functionalities, compare shapes using the overloaded == operator, and calculate the total area using the total Area function.

int main() {

Rectangle rect(3.0, 4.0); // Create rectangle and circle objects

Circle circ(5.0);

vector<Shape\*> shapes; // Create a vector of Shape pointers

shapes.push\_back(&rect);

shapes.push\_back(&circ);

cout << "Rectangle area: " << rect.area() << "\n"; // Test area calculations

cout << "Circle area: " << circ.area() << "\n";

cout << "Rectangles are equal: " << (rect == rect) << "\n"; // Test == operator

cout << "Rectangle and Circle are equal: " << (rect == circ) << "\n";

cout << "Total area: " << totalArea(shapes) << "\n"; // Calculate total area

Point<int> p1(1, 2); // Point template class

Point<double> p2(3.5, 4.5);

p1.display();

p2.display();

return 0;

}

