Create a base class Shape with a pure virtual function draw() that has no implementation. Derive classes Square, Circle, and Triangle from Shape. Each derived class should override draw() to provide its specific drawing behavior (e.g., printing "\*" for square, "OOO" for circle, etc.). Write a function printShape(Shape shape) that takes a base class pointer and calls draw() on it. Demonstrate polymorphism by creating objects of the derived classes, storing them in a Shape\* array, and calling printShape() on each element.

#include <iostream>

#include <vector>

using namespace std;

class Shape {

public:

virtual void draw() const = 0;

virtual ~Shape() {}

};

class Square : public Shape {

public:

void draw() const override {

cout << "\*\*\*\*\n";

cout << "\* \*\n";

cout << "\* \*\n";

cout << "\*\*\*\*\n";

}

};

class Circle : public Shape {

public:

void draw() const override {

cout << " OOO \n";

cout << "O O\n";

cout << " OOO \n";

}

};

class Triangle : public Shape {

public:

void draw() const override {

cout << " \* \n";

cout << " \* \* \n";

cout << "\*\*\*\*\*\n";

}

};

void printShape(const Shape\* shape) {

shape->draw();

}

int main() {

Square square;

Circle circle;

Triangle triangle;

vector<Shape\*> shapes = {&square, &circle, &triangle};

for(const auto& shape : shapes) {

printShape(shape);

cout << endl;

}

return 0;

}

Output:

