Q1.

#include <iostream>

using namespace std;

class Person {

public:

string name;

int age;

void displayInfo() {

cout << "Name: " << name << ", Age: " << age << endl;

}

};

int main() {

Person person1;

person1.name = "John";

person1.age = 25;

Person person2;

person2.name = "Alice";

person2.age = 30;

person1.displayInfo();

person2.displayInfo();

return 0;

}

OUTPUT:

Name: John, Age: 25

Name: Alice, Age: 30

Q2.

#include <iostream>

using namespace std;

class Rectangle {

public:

int length, width;

Rectangle() {

length = 0;

width = 0;

}

Rectangle(int l, int w) {

length = l;

width = w;

}

int calculateArea() {

return length \* width;

}

};

int main() {

Rectangle rect1; // Default constructor

Rectangle rect2(5, 8); // Parameterized constructor

cout << "Area of rect1: " << rect1.calculateArea() << endl;

cout << "Area of rect2: " << rect2.calculateArea() << endl;

return 0;

}

OUTPUT:

Area of rect1: 0

Area of rect2: 40

Q3.

#include <iostream>

using namespace std;

class Shape {

public:

virtual void draw() {

cout << "Drawing a shape." << endl;

}

};

class Circle : public Shape {

public:

void draw() override {

cout << "Drawing a circle." << endl;

}

};

class Square : public Shape {

public:

void draw() override {

cout << "Drawing a square." << endl;

}

};

int main() {

Shape\* shape1 = new Circle();

Shape\* shape2 = new Square();

shape1->draw();

shape2->draw();

delete shape1;

delete shape2;

return 0;

}

OUTPUT:

Drawing a circle.

Drawing a square.

Q4.

#include <iostream>

using namespace std;

class Complex {

public:

double real, imag;

Complex operator+(const Complex& other) {

Complex result;

result.real = real + other.real;

result.imag = imag + other.imag;

return result;

}

};

int main() {

Complex num1, num2, sum;

num1.real = 3;

num1.imag = 5;

num2.real = 2;

num2.imag = 7;

sum = num1 + num2;

cout << "Sum: " << sum.real << " + " << sum.imag << "i" << endl;

return 0;

}

OUTPUT:

Sum: 5 + 12i

Q5.

#include <iostream>

using namespace std;

class Animal {

public:

void eat() {

cout << "Animal is eating." << endl;

}

};

class Dog : public Animal {

public:

void bark() {

cout << "Dog is barking." << endl;

}

};

int main() {

Dog myDog;

myDog.eat(); // Inherited function

myDog.bark(); // Own function

return 0;

}

OUTPUT:

Animal is eating.

Dog is barking.

Q6.

#include <iostream>

using namespace std;

class Student {

public:

string name;

Student (string x){

name = x;

}

// Copy Constructor

Student(const Student& i) {

name = i.name;

}

};

int main() {

Student originalStudent ("Alice");

Student copiedStudent (originalStudent); // Copy constructor is called

cout << "Original Student: " << originalStudent.name << endl;

cout << "Copied Student: " << copiedStudent.name << endl;

return 0;

}

OUTPUT:

Original Student: Alice

Copied Student: Alice

Q7.

public class Person {

String name;

int age;

void displayInfo() {

System.out.println("Name: " + name + ", Age: " + age);

}

public static void main(String[] args) {

Person person1 = new Person();

person1.name = "John";

person1.age = 25;

Person person2 = new Person();

person2.name = "Alice";

person2.age = 30;

person1.displayInfo();

person2.displayInfo();

}

}

OUTPUT:

Name: John, Age: 25Name: Alice, Age: 30

Q8.

import java.applet.Applet;

import java.awt.Graphics;

public class SimpleApplet extends Applet {

public void paint(Graphics g) {

g.drawString("Hello, this is a simple applet!", 20, 20);

}

}

Q9.

import java.awt.\*;

import java.awt.event.\*;

public class AWTWindowDemo {

public static void main(String[] args) {

Frame frame = new Frame("AWT Window Demo");

Button button = new Button("Click me");

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

System.out.println("Button clicked!");

}

});

frame.add(button);

frame.setSize(300, 200);

frame.setLayout(new FlowLayout());

frame.setVisible(true);

frame.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent windowEvent) {

System.exit(0);

}

});

}

}

Q10.

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.ServletException;

import java.io.IOException;

import java.io.PrintWriter;

public class SimpleServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<html><body>");

out.println("<h2>Hello, this is a simple servlet!</h2>");

out.println("</body></html>");

}

}