1. Programs on Operators, Arithmetic Promotion, Method Calling in cpp

**#include <iostream>**

**using namespace std;**

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

**return a + b;**

**}**

**int main() {**

**char x = 'A';**

**short y = 2;**

**int result = add(x, y);**

**cout << result << endl;**

**return 0;**

**}**

2. Programs on dealing with Arrays.

**#include <iostream>**

**using namespace std;**

**int main() {**

**int numbers[5];**

**for (int i = 0; i < 5; i++) {**

**numbers[i] = i;**

**cout << numbers[i] << " ";**

**}**

**cout << endl;**

**return 0;**

**}**

Here's another example of a program that demonstrates how to use a 2D array and how to traverse it:

**#include <iostream>**

**using namespace std;**

**int main() {**

**int rows = 3;**

**int cols = 4;**

**int matrix[rows][cols];**

**int count = 1;**

**for (int i = 0; i < rows; i++) {**

**for (int j = 0; j < cols; j++) {**

**matrix[i][j] = count;**

**count++;**

**cout << matrix[i][j] << " ";**

**}**

**cout << endl;**

**}**

**return 0;**

**}**

1. Programs on Classes: String and Math.

*In C++, the string class is part of the standard library and provides methods for manipulating strings of characters. Some common methods include length(), substr(), and find(). Here is an example program that demonstrates the use of some of these methods:*

**#include <iostream>**

**#include <string>**

**int main() {**

**std::string s = "Hello, world!";**

**std::cout << "The length of the string is: " << s.length() << std::endl;**

**std::cout << "The substring from position 7 to 12 is: " << s.substr(7, 6) << std::endl;**

**std::cout << "The position of the first 'o' in the string is: " << s.find('o') << std::endl;**

**return 0;**

**}**

*The Math class in C++ is not a built-in class, but rather a collection of mathematical functions and constants defined in the cmath header file. Some common functions include sqrt(), pow(), and abs(). Here is an example program that demonstrates the use of some of these functions:*

***#include <iostream>***

***#include <cmath>***

***int main() {***

***double x = -4.5;***

***std::cout << "The square root of " << x << " is " << sqrt(x) << std::endl;***

***std::cout << "2 raised to the power of 3 is " << pow(2, 3) << std::endl;***

***std::cout << "The absolute value of " << x << " is " << abs(x) << std::endl;***

***return 0;***

***}***

1. Programs on Inheritance and Polymorphism
2. Inheritance

**#include <iostream>**

**using namespace std;**

**class Shape {**

**public:**

**void setWidth(int w) {**

**width = w;**

**}**

**void setHeight(int h) {**

**height = h;**

**}**

**protected:**

**int width;**

**int height;**

**};**

**class Rectangle: public Shape {**

**public:**

**int getArea() {**

**return (width \* height);**

**}**

**};**

**class Triangle: public Shape {**

**public:**

**int getArea() {**

**return (width \* height)/2;**

**}**

**};**

**int main() {**

**Rectangle rect;**

**Triangle trgl;**

**rect.setWidth(5);**

**rect.setHeight(7);**

**trgl.setWidth(5);**

**trgl.setHeight(7);**

**cout << "Total area of rectangle: " << rect.getArea() << endl;**

**cout << "Total area of triangle: " << trgl.getArea() << endl;**

**return 0;**

**}**

1. Polymorphism

**#include <iostream>**

**using namespace std;**

**class Shape {**

**public:**

**virtual int area() = 0; // pure virtual function**

**};**

**class Rectangle: public Shape {**

**public:**

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

**width = a;**

**height = b;**

**}**

**int area() {**

**return width \* height;**

**}**

**private:**

**int width;**

**int height;**

**};**

**class Triangle: public Shape {**

**public:**

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

**width = a;**

**height = b;**

**}**

**int area() {**

**return (width \* height)/2;**

**}**

**private:**

**int width;**

**int height;**

**};**

**int main() {**

**Shape \*shape;**

**Rectangle rect(3, 4);**

**Triangle trgl(3, 4);**

**shape = &rect;**

**cout << "Total area of rectangle: " << shape->area() << endl;**

**shape = &trgl;**

**cout << "Total area of triangle: " << shape->area() << endl;**

**return 0;**

**}**

1. **A. Programs on Garbage collection**

**#include <iostream>**

**#include <memory>**

**class MyObject {**

**public:**

**MyObject(int x) : x\_(x) {**

**std::cout << "MyObject created with x = " << x\_ << std::endl;**

**}**

**~MyObject() {**

**std::cout << "MyObject with x = " << x\_ << " destroyed" << std::endl;**

**}**

**int x\_;**

**};**

**int main() {**

**std::unique\_ptr<MyObject> obj1(new MyObject(1));**

**std::unique\_ptr<MyObject> obj2 = std::make\_unique<MyObject>(2);**

**obj1 = std::move(obj2);**

**return 0;**

**}**

1. ***write a program on packaging in cpp***

**#include <iostream>**

**// Define namespace "MyLibrary"**

**namespace MyLibrary {**

**class MyClass {**

**public:**

**void Print() {**

**std::cout << "Hello from MyClass in MyLibrary namespace" << std::endl;**

**}**

**};**

**}**

**int main() {**

**// Create an object of MyClass in the "MyLibrary" namespace**

**MyLibrary::MyClass my\_obj;**

**my\_obj.Print();**

**return 0;**

**}**

1. *write a program on access Modifiers, as well as static and abstract modifiers in cpp.*

example program in C++ that demonstrates the use of the static and abstract modifiers:

**#include <iostream>**

**class Shape {**

**public:**

**// Declare a pure virtual function to make the class abstract**

**virtual double getArea() = 0;**

**};**

**class Rectangle : public Shape {**

**private:**

**double width, height;**

**public:**

**Rectangle(double w, double h) {**

**width = w;**

**height = h;**

**}**

**double getArea() {**

**return width \* height;**

**}**

**};**

**class Circle : public Shape {**

**private:**

**double radius;**

**// Declare a static variable that is shared among all instances of the class**

**static double pi;**

**public:**

**Circle(double r) {**

**radius = r;**

**}**

**double getArea() {**

**return pi \* radius \* radius;**

**}**

**// Declare a static function that can be called without an instance of the class**

**static double getPi() {**

**return pi;**

**}**

**};**

**// Initialize the static variable**

**double Circle::pi = 3.14159265;**

**int main() {**

**Rectangle rect(5, 10);**

**Circle circle(2);**

**std::cout << "Rectangle area: " << rect.getArea() << std::endl;**

**std::cout << "Circle area: " << circle.getArea() << std::endl;**

**std::cout << "Circle pi: " << Circle::getPi() << std::endl;**

**return 0;**

**}**

1. Programs on Interfaces block initializers, final Modifier, as well as static and dynamic binding

***#include <iostream>***

***// Declare an interface (pure virtual class)***

***class Drawable {***

***public:***

***virtual void draw() = 0;***

***};***

***class Shape : public Drawable {***

***protected:***

***double x, y;***

***public:***

***// block initializers***

***Shape(double x, double y) : x(x), y(y) {}***

***virtual void move(double dx, double dy) {***

***x += dx;***

***y += dy;***

***}***

***};***

***class Rectangle : public Shape {***

***private:***

***double width, height;***

***public:***

***// block initializers***

***Rectangle(double x, double y, double w, double h) : Shape(x, y), width(w), height(h) {}***

***void draw() {***

***std::cout << "Drawing a Rectangle at (" << x << ", " << y << ") with width " << width << " and height " << height << std::endl;***

***}***

***};***

***class Circle final : public Shape {***

***private:***

***double radius;***

***public:***

***// block initializers***

***Circle(double x, double y, double r) : Shape(x, y), radius(r) {}***

***void draw() {***

***std::cout << "Drawing a Circle at (" << x << ", " << y << ") with radius " << radius << std::endl;***

***}***

***};***

***// Circle can no longer be subclassed because it has been declared as "final"***

***// class SpecialCircle : public Circle {} // This will produce error***

***int main() {***

***Rectangle rect(1, 2, 3, 4);***

***Circle circle(5, 6, 7);***

***Drawable\* shapes[] = { &rect, &circle };***

***for (auto shape : shapes) {***

***shape->draw(); // dynamic binding***

***}***

***rect.move(1, 1);***

***circle.move(2, 2);***

***for (auto shape : shapes) {***

***shape->draw(); // dynamic binding***

***}***

***return 0;***

***}***

7. Programs on file handling and stream manipulation

***#include <iostream>***

***#include <fstream>***

***int main() {***

***// Open a file for writing***

***std::ofstream outFile("example.txt");***

***// Write some data to the file***

***outFile << "Hello, World!" << std::endl;***

***outFile << 42 << std::endl;***

***outFile << 3.14159 << std::endl;***

***// Close the file***

***outFile.close();***

***// Open the file for reading***

***std::ifstream inFile("example.txt");***

***// Read the data back from the file***

***std::string s;***

***int i;***

***double d;***

***inFile >> s >> i >> d;***

***// Close the file***

***inFile.close();***

***// Print the data to the console***

***std::cout << s << std::endl;***

***std::cout << i << std::endl;***

***std::cout << d << std::endl;***

***return 0;***

***}***

***8 Programs on Dynamic Polymorphism***

***#include <iostream>***

***class Shape {***

***public:***

***virtual double getArea() = 0;***

***};***

***class Rectangle : public Shape {***

***private:***

***double width, height;***

***public:***

***Rectangle(double w, double h) {***

***width = w;***

***height = h;***

***}***

***double getArea() {***

***return width \* height;***

***}***

***};***

***class Circle : public Shape {***

***private:***

***double radius;***

***public:***

***Circle(double r) {***

***radius = r;***

***}***

***double getArea() {***

***return 3.14159265 \* radius \* radius;***

***}***

***};***

***void displayArea(Shape& shape) {***

***std::cout << "Area: " << shape.getArea() << std::endl;***

***}***

***int main() {***

***Rectangle rect(5, 10);***

***Circle circle(2);***

***displayArea(rect);***

***displayArea(circle);***

***return 0;***

***}***

***9. Programs on Dynamic Memory Management***

***#include <iostream>***

***class Shape {***

***public:***

***Shape() {***

***std::cout << "Shape constructed" << std::endl;***

***}***

***virtual ~Shape() {***

***std::cout << "Shape destructed" << std::endl;***

***}***

***virtual double getArea() = 0;***

***};***

***class Rectangle : public Shape {***

***private:***

***double width, height;***

***public:***

***Rectangle(double w, double h) {***

***width = w;***

***height = h;***

***std::cout << "Rectangle constructed" << std::endl;***

***}***

***~Rectangle() {***

***std::cout << "Rectangle destructed" << std::endl;***

***}***

***double getArea() {***

***return width \* height;***

***}***

***};***

***int main() {***

***Shape\* shape = new Rectangle(5, 10);***

***std::cout << "Area: " << shape->getArea() << std::endl;***

***delete shape;***

***return 0;***

***}***

***10. Programs on Exception Handling***

***#include <iostream>***

***using namespace std;***

***int main() {***

***try {***

***int num1, num2;***

***cout << "Enter two numbers: ";***

***cin >> num1 >> num2;***

***if (num2 == 0) {***

***throw "Division by zero exception"; }***

***cout << "The quotient is: " << num1 / num2;***

***} catch (const char\* e) {***

***cout << "Error: " << e << endl;***

***}***

***return 0;***

***}***

11. Programs on generic programming using templates.

***#include <iostream>***

***template <typename T>***

***T add(T a, T b) {***

***return a + b;***

***}***

***int main() {***

***std::cout << add(1, 2) << std::endl; // Outputs 3***

***std::cout << add(1.5, 2.5) << std::endl; // Outputs 4***

***std::cout << add(std::string("Hello "), std::string("World!")) << std::endl; // Outputs "Hello World!"***

***return 0;***

***}***

12. Programs on STL-containers and iterators

***#include <iostream>***

***#include <vector>***

***using namespace std;***

***int main() {***

***vector<int> numbers;***

***// Adding elements to the vector***

***numbers.push\_back(3);***

***numbers.push\_back(4);***

***numbers.push\_back(7);***

***// Accessing elements using an iterator***

***vector<int>::iterator it;***

***for (it = numbers.begin(); it != numbers.end(); ++it) {***

***cout << \*it << " ";***

***}***

***cout << endl;***

***return 0;***

***}***