write a program in C++ to convert seconds into hours , minutes and seconds #include <iostream> using namespace std;

int main() { int seconds, hours, minutes, remaining_seconds;

cout « "Enter the number of seconds: "; cin » seconds;

// Calculating hours hours = seconds / 3600; // Calculating remaining seconds after converting hours remaining_seconds = seconds % 3600; // Calculating minutes minutes = remaining_seconds / 60; // Calculating remaining seconds after converting minutes remaining_seconds = remaining_seconds % 60;

cout « "Hours: " « hours « endl; cout « "Minutes: " « minutes « endl; cout « "Seconds: " « remaining_seconds « endl;

return 0; }

write a C++ program to create a simple calculator

#include <iostream> using namespace std;

int main() { char op; float num1, num2;

cout « ”Enter operator (+, -, *, /): ”; cin » op;

cout « ”Enter two numbers: ”; cin » num1 » num2;

switch(op) { case ’+’: cout « ”Result: ” « num1 + num2; break; case ’-’: cout « ”Result: ” « num1 - num2; break; case ’*’: cout « ”Result: ” « num1 * num2; break; case ’/’: if(num2 != 0) cout « ”Result: ” « num1 / num2; else cout « ”Error! Division by zero!”; break; default: cout « ”Invalid operator!”; }

return 0; }

write a program in C++ using copy constructor to copy data of an object to another object

```
#include <iostream>
using namespace std;

class MyClass {
private:
    int data;

public:
    // Constructor
    MyClass(int d) : data(d) {}

    // Copy constructor
    MyClass(const MyClass &other) {
        data = other.data;
    }

    // Function to display data
    void display() {
        cout << "Data: " << data << endl;
    }
};

int main() {
    // Create an object
    MyClass obj1(5);

    // Create another object and copy data from obj1 using the copy constructor
    MyClass obj2 = obj1;

    // Display data of both objects
    cout << "Object 1:" << endl;
    obj1.display();
    cout << "Object 2:" << endl;
    obj2.display();

    return 0;
}
```

write a program in C++ to generate Fibonacci Series by using Constructor to initialize Data members

```
#include <iostream>
using namespace std;

class Fibonacci {
private:
   int n1, n2; // Previous two Fibonacci numbers

public:
   // Constructor to initialize the first two Fibonacci numbers
   Fibonacci(int first, int second) : n1(first), n2(second) {}

   // Function to generate and print Fibonacci series
   void generateSeries(int terms) {
      int nextTerm;
      cout << "Fibonacci Series:" << endl;
      for (int i = 0; i < terms; ++i) {
         cout << n1 << " ";
         nextTerm = n1 + n2;
         n1 = n2;
         n2 = nextTerm;
      }
      cout << endl;
   }
};

int main() {
   int first, second, terms;

   cout << "Enter the first two numbers of Fibonacci series: ";
   cin >> first >> second;

   cout << "Enter the number of terms: ";
   cin >> terms;

   // Create an object of Fibonacci class with the first two numbers
   Fibonacci fib(first, second);

   // Generate and print the Fibonacci series
   fib.generateSeries(terms);

   return 0;
}
```

write a program in C++ to design a class having a static member function named showcount() which has property of displaying the number of object created of the class

```
#include <iostream>
using namespace std;

class MyClass {
private:
    static int count; // Static member to keep track of the count of objects

public:
    MyClass() {
        count++; // Increment count every time an object is created
    }

    static void showcount() {
        cout << "Number of objects created: " << count << endl;
    }
};

int MyClass::count = 0; // Initialize the static member variable

int main() {
    MyClass obj1, obj2, obj3;

    MyClass::showcount(); // Call static member function to display the count

    MyClass obj4;

    MyClass::showcount(); // Display updated count after creating another object

    return 0;
}
```

write a C++ program illustrating the use of virtual functions in class

#include <iostream> using namespace std;

// Base class class Shape { public: // Virtual function to calculate area virtual double calculateArea() { return 0; } };

// Derived class 1 class Rectangle : public Shape { private: double length; double width;

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

// Override virtual function to calculate area of rectangle double calculateArea() override { return length * width; } };

// Derived class 2 class Circle : public Shape { private: double radius;

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

// Override virtual function to calculate area of circle double calculateArea() override { return 3.14 * radius * radius; } };

int main() { // Create objects of Rectangle and Circle Rectangle rectangle(5, 4); Circle circle(3);

// Call virtual function to calculate area cout « "Area of Rectangle: " « rectangle.calculateArea() « endl; cout « "Area of Circle: " « circle.calculateArea() « endl;

return 0; }

write a C++ program to maintain the records of person with details (name and age) and find the eldest among them. the program must be this pointer to return the result

#include <iostream> #include <string> using namespace std;

class Person { private: string name; int age;

public: // Constructor Person(string n, int a) : name(n), age(a) {}

// Function to compare age and return the eldest person using this pointer Person& eldest(Person& other) { if (this->age >= other.age) return *this; else return other; }

// Function to display person details void display() { cout « "Name: " « name « ", Age: " « age « endl; } };

int main() { // Create Person objects Person person1("John", 30); Person person2("Alice", 25); Person person3("Bob", 35);

// Find the eldest person Person& eldestPerson = person1.eldest(person2).eldest(person3);

// Display the eldest person cout « "The eldest person is: "; eldestPerson.display();

return 0; }

write a program in C++ to design a class representing complex numbers and having the functionality of performing addition and multiplication of two complex numbers using operator overloading

#include <iostream> using namespace std;

class Complex { private: double real; double imag;

public: // Constructor Complex(double r = 0, double i = 0) : real(r), imag(i) {}

// Overloaded addition operator Complex operator+(const Complex& other) const { return Complex(real + other.real, imag + other.imag); }

// Overloaded multiplication operator Complex operator*(const Complex& other) const { return Complex((real * other.real) - (imag * other.imag), (real * other.imag) + (imag * other.real)); }

// Function to display complex number void display() const { cout « "(" « real « " + " « imag « "i)" « endl; } };

int main() { // Create complex numbers Complex c1(2, 3); Complex c2(4, 5);

// Perform addition and display result cout « "Addition result: "; Complex add_result = c1 + c2; add_result.display();

// Perform multiplication and display result cout « "Multiplication result: "; Complex mul_result = c1 * c2; mul_result.display();

return 0; }

write a program in C++ using class and object student to print name of the student , roll_no display the same

#include <iostream> using namespace std;

// Class definition class Student { private: string name; int roll_no;

public: // Constructor to initialize name and roll number Student(string n, int r) { name = n; roll_no = r; }

// Function to display student details void display() { cout « "Name: " « name « endl; cout « "Roll Number: " « roll_no « endl; } };

int main() { // Creating an object of the Student class Student student1("John Doe", 12345);

// Displaying student details student1.display();

return 0; }

write a program in C++ to find the greatest of 3 numbers

#include <iostream> using namespace std;

int main() { double num1, num2, num3;

cout « "Enter three numbers: "; cin » num1 » num2 » num3;

if (num1 >= num2 && num1 >= num3) cout « "The greatest number is: " « num1 « endl; else if (num2 >= num1 && num2 >= num3) cout « "The greatest number is: " « num2 « endl; else cout « "The greatest number is: " « num3 « endl;

return 0; }

write a program in C++ to find the sum of even and odd and natural numbers

#include <iostream> using namespace std;

int main() { int limit; cout « "Enter the limit: "; cin » limit;

int sum_even = 0, sum_odd = 0, sum_natural = 0;

// Calculate the sum of even, odd, and natural numbers for (int i = 1; i <= limit; ++i) { sum_natural += i; if (i % 2 == 0) sum_even += i; else sum_odd += i; }

// Output the results cout « "Sum of even numbers up to " « limit « ": " « sum_even « endl; cout « "Sum of odd numbers up to " « limit « ": " « sum_odd « endl; cout « "Sum of natural numbers up to " « limit « ": " « sum_natural « endl;

return 0; }

write a program in C++ to find the volume of a square , cone and rectangle #include <iostream> #include <cmath> using namespace std;

const double PI = 3.14159265358979323846;

// Function to calculate the volume of a square-based pyramid double squarePyramidVolume(double base_length, double height) { return (base_length * base_length * height) / 3.0; }

// Function to calculate the volume of a cone double coneVolume(double radius, double height) { return (PI * radius * radius * height) / 3.0; }

// Function to calculate the volume of a rectangular prism double rectangularPrismVolume(double length, double width, double height) { return length * width * height; }

int main() { double base_length, height, radius, length, width;

// Input for square-based pyramid cout « "Enter base length and height of the square-based pyramid: "; cin » base_length » height; cout « "Volume of the square-based pyramid: " « squarePyramidVolume(base_length, height) « endl;

// Input for cone cout « "Enter radius and height of the cone: "; cin » radius » height; cout « "Volume of the cone: " « coneVolume(radius, height) « endl;

// Input for rectangular prism cout « "Enter length, width, and height of the rectangular prism: "; cin » length » width » height; cout « "Volume of the rectangular prism: " « rectangularPrismVolume(length, width, height) « endl;

return 0; }