File: 1.cpp

```
// Question 1. Write a program to take name, address as character array, age as int , salary as float
//      and contains inline functions to set the values and display it.
#include<iostream>
using namespace std;
class Emp {
    char name[50], address[100];
    int age;
    float salary;
public:
    inline void set() {
        cout<<"Enter Name, Address, age,salary :" <<endl;
        cin >> name >> address >> age >> salary;
    }
    inline void show() {
        cout<<"Name : "<<name<<endl;
        cout<<"Address :"<<address<<endl;
        cout<<"Age :"<<age<<endl;
        cout<<"Salary : "<<salary<<endl;
    }
};
int main() {
    Emp e;
    e.set();
    e.show();
    return 0;
}
```

File: 10.cpp

```
// 10. Write a program to perform addition of two complex numbers using constructor overloading. The first
//       constructor which takes no argument is used to create objects which are not initialized, second which
//     takes one argument is used to initialize real and imag parts to equal values and third which takes two
//     argument is used to initialized real and imag to two different values.
#include <iostream>
using namespace std;
class Complex {
private:
    float real, imag;
public:
    Complex() {
        real = imag = 0;
    }
    Complex(float value) {
        real = imag = value;
    }
    Complex(float r, float i) {
        real = r;
        imag = i;
    }
    Complex add(Complex c) {
        Complex result;
        result.real = real + c.real;
        result.imag = imag + c.imag;
        return result;
    }
    void display() {
        cout << real << " + " << imag << "i" << endl;
    }
};
int main() {
    float both, r, i;
    cout<<"Enter 1 value for both and 1 real,1 imaginary value :"<<endl;
    cin >> both >> r >> i;
    Complex c1(both);
    Complex c2(r, i);
    Complex c3 = c1.add(c2);
    cout << "Sum of complex numbers: ";
    c3.display();
    return 0;
}
```

File: 11.cpp

```
// 11. Write a program to generate a Fibonacci series using copy constructor.
#include <iostream>
using namespace std;

class Fibonacci {
    int a, b, n;
public:
    Fibonacci(int num) {
        a = 0;
        b = 1;
        n = num;
        if (n > 2)
            cout << a << " " << b << " ";
    }

    Fibonacci(Fibonacci &f) {
        int a1 = f.a, b1 = f.b, c;
        for (int i = 2; i < f.n; i++) {
            c = a1 + b1;
            cout << c << " ";
            a1 = b1;
            b1 = c;
        }
    }
};

int main() {
    int num;
    cout << "Enter number of terms more than 2: ";
    cin >> num;
    Fibonacci f1(num);
    Fibonacci f2(f1);
    return 0;
}
```

File: 12.cpp

```
// 12. Create a class which keep track of number of its instances. Use static data member, constructors and
//     destructors to maintain updated information about active objects
#include <iostream>
using namespace std;
class Tracker {
private:
    static int count;
public:
    Tracker() {
        count++;
        cout << "Object created. Active objects: " << count << endl;
    }
    ~Tracker() {
        count--;
        cout << "Object destroyed. Active objects: " << count << endl;
    }
    static void showCount() {
        cout << "Currently active objects: " << count << endl;
    }
};
int Tracker::count = 0;
int main() {
    Tracker::showCount();
    Tracker a,b;
    Tracker::showCount();
    Tracker c;
    Tracker::showCount();
    return 0;
}
```

File: 13.cpp

```
// 13. Write a program to demonstrate the use of this pointer.
#include <iostream>
using namespace std;
class Student {
private:
    int roll;
    string name;
public:
    void setData(int roll, string name) {
        this->roll = roll;
        this->name = name;
    }
    void display() {
        cout << "Name: " << this->name << endl;
        cout << "Roll No: " << this->roll << endl;
    }
};
int main() {
    Student s;
    int roll;
    string name;
    cout << "Enter roll number and name: ";
    cin >> roll >> name;
    s.setData(roll, name);
    s.display();
    return 0;
}
```

File: 14.cpp

```
// 14. Write a program to find the biggest of three numbers using friend function
#include <iostream>
using namespace std;
class Number {
private:
    int num;
public:
    Number(int n) {
        num = n;
    }
    friend void findBiggest(Number, Number, Number);
};
void findBiggest(Number a, Number b, Number c) {
    int max;
     if (a.num >= b.num && a.num >= c.num)
        max = a.num;
    else if (b.num >= a.num && b.num >= c.num)
        max = b.num;
    else
        max = c.num;
    cout << "Biggest number is: " << max << endl;
}
int main() {
    int x, y, z;
    cout << "Enter three numbers: ";
    cin >> x >> y >> z;
    Number n1(x), n2(y), n3(z);
    findBiggest(n1, n2, n3);
    return 0;
}
```

File: 15.cpp

```
// 15. Write a program to demonstrate the use of friend function with Inline assignment
#include <iostream>
using namespace std;
class Number {
private:
    int value;
public:
    Number(int v) {
        value = v;
    }
    friend void inline displayDouble(Number);
};
void inline displayDouble(Number n) {
    cout << "Double: " << n.value * 2 << endl;
}
int main() {
    int x;
    cout << "Enter a number: ";
    cin >> x;
    Number num(x);
    displayDouble(num);
    return 0;
}
```

File: 16.cpp

```
// 16. Write a program to find the greatest of two given numbers in two different classes using friend function.
#include <iostream>
using namespace std;
class B;
class A {
    int num;
public:
    A(int n) { num = n; }
    friend void findGreatest(A, B);
};
class B {
    int num;
public:
    B(int n) { num = n; }
    friend void findGreatest(A, B);
};
void findGreatest(A a, B b) {
    if (a.num > b.num)
        cout << "Greatest: " << a.num << endl;
    else
        cout << "Greatest: " << b.num << endl;
}
int main() {
    int x, y;
    cout << "Enter Two Values ";
    cin >> x >> y;
    A obj1(x);
    B obj2(y);
    findGreatest(obj1, obj2);
    return 0;
}
```

File: 17.cpp

```
// 17. Write a program to find the sum of two numbers declared in a class and display the numbers and sum using friend class.
#include <iostream>
using namespace std;
class Numbers {
    int a, b;
public:
    Numbers(int x, int y) {
        a = x;
        b = y;
    }
    friend class Sum;
};
class Sum {
public:
    void display(Numbers n) {
        int s = n.a + n.b;
        cout << "Sum: " << s << endl;
    }
};
int main() {
    int x, y;
    cout << "Enter two numbers: ";
    cin >> x >> y;
    Numbers obj(x, y);
    Sum s;
    s.display(obj);
    return 0;
}
```

File: 18.cpp

```
// 18. Write a program to overload unary increment (++) operator .
#include <iostream>
using namespace std;
class Number {
    int value;
public:
    Number(int v) {
        value = v;
    }
    // returns the modified object
    Number operator++() {
        ++value;
        return *this;
    }
    void display() {
        cout << "Value: " << value << endl;
    }
};
int main() {
    int x;
    cout << "Enter a number: ";
    cin >> x;
    Number n(x);
    ++n;            // Pre-increment
    n.display();
    return 0;
}
```

File: 19.cpp

```
// 19. Write a program to overload binary + operator
#include <iostream>
using namespace std;
class Number {
    int value;
public:
    Number(int v) {
        value = v;
    }
    // Overload binary + operator
    Number operator+(Number obj) {
        return Number(value + obj.value);
    }
    void display() {
        cout << "Sum of the two numbers: " << value << endl;
    }
};
int main() {
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    Number n1(a);
    Number n2(b);
    Number n3 = n1 + n2; // Calls overloaded + operator
    n3.display();
    return 0;
}
```

File: 2.cpp

```
// 2. Using the concept of function overloading Write function for calculating
//     the area of triangle ,circle and rectangle.
#include <iostream>
using namespace std;
class Shape {
    public:
    float area(float b, float h) {
        return 0.5 * b * h;
    }
    float area(float r) {
        return 3.14 * r * r;
    }
    double area(double l, double b) {
        return l * b;
    }
};
int main() {
    Shape s;
    float base , height, radius;
    double length, breadth;
    cout<<"Enter Base, Height, Radius, Length , Breath,";
    cin >> base >> height >> radius >> length >> breadth;
    cout <<"Area of Triangle"<< s.area(base, height) << endl ;
    cout<< "Area of Circle " <<s.area(radius) << endl;
    cout<< "Area of Rectangle " << s.area(length, breadth) << endl;
    return 0;
}
```

File: 20.cpp

```
// 20. Write a program to overload less than (<) operator
#include <iostream>
using namespace std;
class Number {
    int value;
public:
    Number(int v) {
        value = v;
    }
    // Overload less than operator (<)
    bool operator<(Number obj) {
        return value < obj.value;
    }
    void display() {
        cout << "Greater Value: " << value << endl;
    }
};
int main() {
    int a, b;
    cout << "Enter two number: ";
    cin >> a >> b;
    Number n1(a);
    Number n2(b);
    if (n1 < n2)
        n2.display();
    else
        n1.display();

    return 0;
}
```

File: 21.cpp

```cpp
// 21. Write a program to overload assignment (=) operator.
#include <iostream>
using namespace std;
class Number {
    int value;
public:
    Number(int v) {
        value = v;
    }
    // Overload assignment operator
    Number operator=(Number obj) {
        if (this != &obj) {  // Check for self-assignment
            value = obj.value;
        }
        return *this;
    }
    void display() {
        cout << "Value: " << value << endl;
    }
};
int main() {
    int a, b;
    cout << "Enter two number: ";
    cin >> a >> b;
    Number n1(a);
    Number n2(b);
    cout << "Before assignment:" << endl;
    n1.display();
    n2.display();
    // Overloaded assignment operator
    n1 = n2;
    cout << "\nAfter assignment:" << endl;
    n1.display();
    n2.display();
    return 0;
}
```

File: 23.cpp

```
// 23. Write a program to overload new and delete operators.
#include <iostream>
using namespace std;
class Sample {
public:
    void* operator new(size_t size) {
        cout << "Custom new called\n";
        void* p = malloc(size);
        return p;
    }

    void operator delete(void* p) {
        cout << "Custom delete called\n";
        free(p);
    }
};

int main() {
    Sample* obj = new Sample;
    delete obj;
    return 0;
}
```

File: 24.cpp

```
// 24. Write a program to overload unary minus (-) operator using friend function.
#include <iostream>
using namespace std;
class Number {
    int value;
public:
    Number(int v = 0) {
        value = v;
    }
    friend Number operator-(Number obj);

    void display() {
        cout << "Value: " << value << endl;
    }
};
Number operator-(Number obj) {
    return Number(-obj.value);
}
int main() {
    int n;
    cout<< "Enter any number: ";
    cin >> n;
    Number num(n);
    cout << "Original ";
    num.display();
    Number negNum = -num; // Using the overloaded unary - operator
    cout << "After applying unary minus: ";
    negNum.display();
    return 0;
}
```

File: 25.cpp

```
// 25. Create a base class basic_info with data members name ,roll no, sex and two member functions getdata
// and display. Derive a class physical_fit from basic_info which has data members height and weight and member
//  functions getdata and display. Display all the information using object of derived class.
#include <iostream>
using namespace std;
class basic_info {
protected:
    string name;
    int roll_no;
    char sex;
public:
    void getdata() {
        cout << "Enter name, roll number, sex(M/F) ";
        cin >> name >> roll_no >> sex;
    }
    void display() {
         cout << "Name: " << name << endl << "Roll Number: " << roll_no << endl << "Sex: " << sex << endl;
    }
};
class physical_fit : public basic_info {
private:
    float height;
    float weight;

public:
    void getdata() {
        basic_info::getdata();
        cout << "Enter height (in cm), weight (in kg) ";
        cin >> height >> weight;
    }

    void display() {
        basic_info::display();
         cout << "Height: " << height << " cm" << endl << "Weight: " << weight << " kg" << endl;
    }
};
int main() {
    physical_fit student;
    student.getdata();
    student.display();
    return 0;
}
```

File: 27.cpp

```
// 27. Design three classes STUDENT ,EXAM and RESULT. The STUDENT class has datamembers such as rollno, name.
// create a class EXAM by inheriting the STUDENT class. The EXAM class adds datamembers representing the marks
// scored in six subjects. Derive the RESULT from the EXAM class and has its own datamembers such as totalmarks.
// Write a program to model this relationship.
#include <iostream>
using namespace std;
class STUDENT {
protected:
    int rollno;
    string name;
public:
    void getdata() {
        cout << "Enter roll number: ";
        cin >> rollno;
        cin.ignore();
        cout << "Enter name: ";
        getline(cin, name);
    }
    void putdata() {
        cout << "Roll Number: " << rollno << endl;
        cout << "Name: " << name << endl;
    }
};
class EXAM : public STUDENT {
protected:
    int marks[6];
public:
    void getdata() {
        STUDENT::getdata();
        cout << "Enter marks for 6 subjects: ";
        for (int i = 0; i < 6; i++) {
            cin >> marks[i];
        }
    }
    void putdata() {
        STUDENT::putdata();
        for (int i = 0; i < 6; i++) {
            cout << "Subject " << i + 1 << " Marks: " << marks[i] << endl;
        }
    }
};
class RESULT : public EXAM {
private:
    int totalmarks;
public:
    void getdata() {
        EXAM::getdata();
```

```
        totalmarks = 0;
        for (int i = 0; i < 6; i++) {
            totalmarks += marks[i];
        }
    }
    void putdata() {
        EXAM::putdata();
        cout << "Total Marks: " << totalmarks << endl;
    }
};
int main() {
    RESULT student;
    student.getdata();
    cout << "\nStudent Result:\n";
    student.putdata();
    return 0;
}
```

File: 29.cpp

```
// 29. Create a base class called SHAPE. Use this class to store two double type values. Derive two specific
// classes called TRIANGLE and RECTANGLE from the base class. Add to the base class, a member function getdata
// to initialize base class datamembers and another member function display to compute and display the area of
//  figures. Make display a virtual function and redefine this function in the derived classes to suit their
//  requirements. Using these three classes design a program that will accept driven of a TRINGLE or RECTANGLE
//  interactively and display the area.
#include <iostream>
using namespace std;
class SHAPE {
protected:
    double x, y;
public:
    void getdata() {
        cout << "Enter two dimensions: ";
        cin >> x >> y;
    }
    virtual void display() {
        cout << "Base SHAPE display\n";
    }
    virtual ~SHAPE() {}
};
class TRIANGLE : public SHAPE {
public:
    void display() {
        double area = 0.5 * x * y;
        cout << "Area of Triangle: " << area << endl;
    }
};
class RECTANGLE : public SHAPE {
public:
    void display() {
        double area = x * y;
        cout << "Area of Rectangle: " << area << endl;
    }
};
int main() {
    SHAPE* shape;
    int choice;
    cout << "Choose shape:\n1. Triangle\n2. Rectangle\nEnter choice: ";
    cin >> choice;
    if (choice == 1) {
        shape = new TRIANGLE;
    } else if (choice == 2) {
        shape = new RECTANGLE;
    } else {
```

```
        cout << "Invalid choice.";
        return 0;
    }
    shape->getdata();
    shape->display();
    delete shape;
    return 0;
}
```

File: 3.cpp

```
// 3. Write a function power to raise a number m to power n. The function takes a double value for m and int value for n.
//       Use default value for n to make the function to calculate squares when this argument is omitted.
#include <iostream>
#include <cmath>
using namespace std;
class PowerCalculator {
public:
    double power(double m, int n = 2) {
        return pow(m, n);
    }
};
int main() {
    PowerCalculator pc;
    double base;
    int exponent;
    cout << "Enter a number: ";
    cin >> base;
    cout << "Enter exponent (enter -1 to skip): ";
    cin >> exponent;
    if (exponent == -1)
        cout << "Result: " << pc.power(base) << endl;
    else
        cout << "Result: " << pc.power(base, exponent) << endl;
    return 0;
}
```

File: 4.cpp

```
// 4. Create a class TIME with members hours, minutes, seconds. Take input, add two time objects
//      passing objects to function and display result.
#include <iostream>
using namespace std;
class TIME {
public:
    int hours, minutes, seconds;
    void getTime() {
        cout << "Enter hours, minutes, and seconds: ";
        cin >> hours >> minutes >> seconds;
    }
    TIME addTime(TIME t) {
        TIME result;
        result.seconds = seconds + t.seconds;
        result.minutes = minutes + t.minutes + result.seconds / 60;
        result.seconds %= 60;
        result.hours = hours + t.hours + result.minutes / 60;
        result.minutes %= 60;
        return result;
    }
    void displayTime() {
        cout << "Time: " << hours << "h " << minutes << "m " << seconds << "s" << endl;
    }
};
int main() {
    TIME t1, t2, t3;
    t1.getTime();
    t2.getTime();
    t3 = t1.addTime(t2);
    t3.displayTime();
    return 0;
}
```

File: 5.cpp

```cpp
// 5. Write a program for multiplication of two matrices using OOP
#include <iostream>
using namespace std;
class Matrix {
public:
    int mat[100][100];
    int rows, cols;
    void input() {
        cout << "Enter rows and columns: ";
        cin >> rows >> cols;
        cout << "Enter elements:\n";
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                cin >> mat[i][j];
            }
        }
    }
    void display() {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                cout << mat[i][j] << " ";
            }
            cout << endl;
        }
    }
    Matrix multiply(Matrix m) {
        Matrix result;
        if (cols != m.rows) {
            cout << "Matrix multiplication not possible.\n";
            result.rows = result.cols = 0;
            return result;
        }
        result.rows = rows;
        result.cols = m.cols;
        for (int i = 0; i < result.rows; i++) {
            for (int j = 0; j < result.cols; j++) {
                result.mat[i][j] = 0;
                for (int k = 0; k < cols; k++) {
                    result.mat[i][j] += mat[i][k] * m.mat[k][j];
                }
            }
        }
        return result;
    }
};
int main() {
    Matrix m1, m2, result;
    cout << "Enter first matrix:\n";
    m1.input();
    cout << "Enter second matrix:\n";
```

```
    m2.input();
    result = m1.multiply(m2);
    if (result.rows > 0 && result.cols > 0) {
        cout << "Resultant matrix:\n";
        result.display();
    }
    return 0;
}
```

File: 6.cpp

```
// 6. Create a class Student which has data members as name, branch, roll no, age ,sex ,marks in five subjects.
//       Display the name of the student and his percentage who has more than 70%.Use array of objects.
#include <iostream>
using namespace std;
class Student {
public:
    string name, branch;
    int rollNo, age;
    char sex;
    int marks[5];
    void input() {
        cout << "Enter name, branch, roll no, age, sex (M/F): ";
        cin >> name >> branch >> rollNo >> age >> sex;
        cout << "Enter marks in 5 subjects: ";
        for (int i = 0; i < 5; i++) {
            cin >> marks[i];
        }
    }
    float calculatePercentage() {
        int total = 0;
        for (int i = 0; i < 5; i++) {
            total += marks[i];
        }
        return total / 5.0;
    }
    void displayIfAbove70() {
        float percentage = calculatePercentage();
        if (percentage > 70) {
            cout << "Name: " << name << ", Percentage: " << percentage << "%" << endl;
        }
    }
};
int main() {
    int n;
    cout << "Enter number of students: ";
    cin >> n;
    Student s[100];
    for (int i = 0; i < n; i++) {
        cout << "\nEnter details for student " << i + 1 << ":\n";
        s[i].input();
    }
    cout << "\nStudents with more than 70%:\n";
    for (int i = 0; i < n; i++) {
        s[i].displayIfAbove70();
    }
    return 0;
}
```

File: 7.cpp

```
// 7. Write a program access members of a student class using pointer to object members
(or using indirection operator).
#include <iostream>
using namespace std;
class Student {
public:
    string name;
    int rollNo;
    float marks;
    void input() {
        cout << "Enter name, roll no and marks: ";
        cin >> name >> rollNo >> marks ;
    }
    void display() {
         cout << "Name: " << name << endl << "Roll No: " << rollNo << endl << "Marks: "
<< marks << endl;
    }
};
int main() {
    Student s;
    Student* ptr = &s;
    ptr->input();
    cout << "\nStudent Details:\n";
    ptr->display();
    return 0;
}
```

File: 9.cpp

```
// 9. Write a program to enter any number and find its factorial  using constructor.
#include <iostream>
using namespace std;
class Factorial {
private:
    int num;
    unsigned long long fact;
public:
    Factorial(int n) {
        num = n;
        fact = 1;
        for (int i = 1; i <= num; i++) {
            fact *= i;
        }
    }
    void display() {
        cout << "Factorial of " << num << " is " << fact << endl;
    }
};
int main() {
    int number;
    cout << "Enter a number: ";
    cin >> number;
    Factorial f(number);
    f.display();
    return 0;
}
```