1. Write a program that finds the volume of different shapes (like rectangle, cylinder, cube) using function overloading.

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

// function to find the volume of a rectangle
double findVolume(double length, double width, double height) {
    return length * width * height;
}

// function to find the volume of a cylinder
double findVolume(double radius, double height) {
    return M_PI * radius * radius * height;
}

// function to find the volume of a cube
double findVolume(double side) {
    return side * side * side;
}

int main() {
    double l, w, h, r, rec_area, cyl_area, cub_area;

    cout << "Enter the length, width and height of rectangle: ";
    cin >> l >> w >> h;
    rec_area = findVolume(l, w, h);

    cout << "Enter the radius and height of cylinder: ";
    cin >> r >> h;
    cyl_area = findVolume(r, h);

    cout << "Enter the length of a side of a cube: ";
    cin >> l;
    cyl_area = findVolume(l);

    cout << "Volume of rectangle: " << rec_area << endl;
    cout << "Volume of cylinder: " << cyl_area << endl;
    cout << "Volume of cube: " << cyl_area << endl;
    return 0;
}
```

2. Write a class to represent time that includes the member function to perform the following:

a. Take input for time in hours and minutes.
b. Add two times.
c. Display the time in form hours: minutes.

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

class Time
{
private:
    int hours;
    int minutes;

public:
    // function to take input for time in hours and minutes
    void input()
    {
        cout << "Enter time in hours and minutes: ";
        cin >> hours >> minutes;
    }

    // function to add two times
    Time operator+(const Time &other)
    {
        Time sum;
        sum.minutes = minutes + other.minutes;
        sum.hours = hours + other.hours + sum.minutes / 60;
        sum.minutes %= 60;
        return sum;
    }

    // function to display the time in form hours:minutes
    void display() const
    {
        cout << hours << ":" << minutes << endl;
    }
};

int main()
{
    Time t1, t2, t3;
    t1.input();
    t2.input();
    t3 = t1 + t2;
    cout << "Sum of two time values (hours:minutes): ";
    t3.display();
```

```
    return 0;
}
```

3. Create a class called COMPLEX that has two private data called real and imaginary. Include constructor function to input real & imaginary values, show() to display complex numbers. Write a program to add two complex numbers.

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

class Complex
{
private:
    double real;
    double imaginary;

public:
    // constructor to initialize real and imaginary
    Complex(double r, double i) : real(r), imaginary(i) {}

    // default constructor
    Complex(){};

    // function to display complex numbers
    void show() const
    {
        cout << real << " + " << imaginary << "i" << endl;
    }

    // function to add two complex numbers
    Complex operator+(const Complex &other)
    {
        Complex sum;
        sum.real = real + other.real;
        sum.imaginary = imaginary + other.imaginary;
        return sum;
    }
};

int main()
{
    Complex c1(2, 3);
    Complex c2(1, 2);
    Complex c3 = c1 + c2;
    cout << "Sum of c1 and c2: ";
    c3.show();
```

```
    return 0;
}
```

4. Write a program to find the largest and smallest number between two numbers of different classes.

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

class A
{
private:
    int x;

public:
    // constructor to initialize x
    A(int a){
        x = a;
    }

    // function to return x
    int getX() const
    {
        return x;
    }
};

class B
{
private:
    double y;

public:
    // constructor to initialize y
    B(double b){
        y = b;
    }

    // function to return y
    double getY() const
    {
        return y;
```

```
    }
};

int main()
{
    A a1(5);
    A a2(10);

    // find largest and smallest numbers between a1 and a2
    int min_a = min(a1.getX(), a2.getX());
    int max_a = max(a1.getX(), a2.getX());
    cout << "Largest number between a1 and a2: " << max_a << endl;
    cout << "Smallest number between a1 and a2: " << min_a << endl;

    return 0;
}
```

5. Using class write a program that would be able to do the following task:
    a. To create the vector.
    b. To add the value of a two vector
    c. To modify the value of a given element
    d. To display the vector in the form (ai+bj+ck)

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

class Vector
{
private:
    vector<double> v;

public:
    // constructor to initialize the vector
    Vector(int size)
    {
        v = vector<double>(size);
    }

    // function to add the values of two vectors
    Vector operator+(const Vector &other)
    {
        Vector sum(v.size());
        for (int i = 0; i < v.size(); i++)
        {
            sum.v[i] = v[i] + other.v[i];
```

```
        }
        return sum;
    }

    // function to modify the value of a given element
    void modify(int index, double value)
    {
        v[index] = value;
    }

    // function to display the vector in the form (ai + bj + ck)
    void display() const
    {
        cout << "(";
        for (int i = 0; i < v.size(); i++)
        {
            cout << v[i] << "i";
            if (i != v.size() - 1)
            {
                cout << " + ";
            }
        }
        cout << ")" << endl;
    }
};

int main()
{
    Vector v1(3);
    v1.modify(0, 1);
    v1.modify(1, 2);
    v1.modify(2, 3);

    cout << "Vector 1: ";
    v1.display();

    Vector v2(3);
    v2.modify(0, 4);
    v2.modify(1, 5);
    v2.modify(2, 6);

    cout << "Vector 2: ";
    v2.display();

    Vector v3 = v1 + v2;
    cout << "Sum of v1 and v2: ";
```

```
    v3.display();
    return 0;
}
```

6. Create a class float that contains one float data number. Overload all the four arithmetic operators for two objects.

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

class Float {
private:
  float x;

public:
  // constructor to initialize x
  Float(float a) : x(a) {}

  // function to return x
  float getX() const {
    return x;
  }

  // overload the + operator for two Float objects
  Float operator+(const Float& other) {
    return Float(x + other.x);
  }

  // overload the - operator for two Float objects
  Float operator-(const Float& other) {
    return Float(x - other.x);
  }

  // overload the * operator for two Float objects
  Float operator*(const Float& other) {
    return Float(x * other.x);
  }

  // overload the / operator for two Float objects
  Float operator/(const Float& other) {
    return Float(x / other.x);
  }
};

int main() {
```

```
  Float f1(2.5);
  Float f2(1.5);
  Float f3 = f1 + f2;
  cout << "Sum of f1 and f2: " << f3.getX() << endl;
  Float f4 = f1 - f2;
  cout << "Difference of f1 and f2: " << f4.getX() << endl;
  Float f5 = f1 * f2;
  cout << "Product of f1 and f2: " << f5.getX() << endl;
  Float f6 = f1 / f2;
  cout << "Quotient of f1 and f2: " << f6.getX() << endl;
  return 0;
}
```

7. Given the following base class:
   class area_cl
   {
   public:
   double height; double width;
   }
   Create two derived classes called rectangle and isosceles that inherit area_cl. Have each class include a function area() that returns the area of a rectangle isosceles triangle, as appropriate. Use parameterized constructors to initialize height and width. Write the complete program.

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

class area_cl
{
public:
    double height;
    double width;
};

class rectangle : public area_cl
{
public:
    // constructor to initialize height and width
    rectangle(double h, double w)
    {
        height = h;
        width = w;
    }

    // function to return the area of the rectangle
    double area()
    {
```

```
        return height * width;
    }
};

class isosceles : public area_cl
{
public:
    // constructor to initialize height and width
    isosceles(double h, double w)
    {
        height = h;
        width = w;
    }

    // function to return the area of the isosceles triangle
    double area()
    {
        return 0.5 * height * width;
    }
};

int main()
{
    rectangle r(2, 3);
    cout << "Area of the rectangle: " << r.area() << endl;
    isosceles t(3, 4);
    cout << "Area of the isosceles triangle: " << t.area() << endl;
    return 0;
}
```

8. Create an abstract base class called shape. Derive class rectangle from the base class shape and a class cube from the rectangle class.

   Data members:
   length, width for class rectangle.
   height for class cube.

   Member function:
   area(), print() for class rectangle.
   volume(), print() for class cube.

   Make function print() as virtual and declare as a pure virtual function in the base class. Write a program to compute the area of rectangle and volume of cube and display the result using base class pointer.

```
#include <iostream>
```

```
using namespace std;
class shape
{
public:
    int length, width, height;
    virtual void area()
    {
        cout << "Shape Area" << endl;
    }
    virtual void volume()
    {
        cout << "Shape Volume" << endl;
    }
    virtual void print() = 0;
};
class rectangle : public shape
{
    int areas;

public:
    void setValue(int l, int w)
    {
        length = l;
        width = w;
    }
    void area()
    {
        areas = length * width;
    }
    void print()
    {
        cout << "The area of a rectangle is: " << areas << endl;
    }
};
class cube : public rectangle
{
    int areas;

public:
    void setValue(int a)
    {
        height = a;
    }
    void volume()
    {
        areas = height * height * height;
```

```
    }
    void print()
    {
        cout << "The volume of a cube is: " << areas << endl;
    }
};
int main()
{
    shape *sp;
    rectangle r;
    r.setValue(4, 6);
    sp = &r;
    sp->area();
    sp->print();

    cube c;
    c.setValue(5);
    sp = &c;
    sp->volume();
    sp->print();

    return 0;
}
```

9. Write a program to read two files simultaneously.

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream file1, file2;
    string line1, line2;

    // Opening files in input mode
    file1.open("file1.txt");
    file2.open("file2.txt");

    // Checking if files are open
    if (!file1.is_open() || !file2.is_open())
    {
        cout << "Error opening files!" << endl;
```

```
        return 0;
    }

    // Reading and printing lines from both files simultaneously
    while (getline(file1, line1) && getline(file2, line2))
    {
        cout << "File1: " << line1 << endl;
        cout << "File2: " << line2 << endl;
    }

    // Closing files
    file1.close();
    file2.close();

    return 0;
}
```

10. Write a program to perform the deletion of white spaces such as horizontal tab, vertical tab, space. line feed, new line and carriage return from a text file and to store the contents of the file without white spaces on another file.

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

int main()
{
    // Open the input file
    ifstream inFile;
    inFile.open("input.txt");

    // Open the output file
    ofstream outFile;
    outFile.open("output.txt");

    // Read the input file character by character
    char ch;
    while (inFile.get(ch))
    {
        // If the character is not a white space, write it to the output file
        if (ch != ' ' && ch != '\t' && ch != '\n' && ch != '\r')
        {
            outFile << ch;
        }
    }
```

```
    // Close the input and output files
    inFile.close();
    outFile.close();

    return 0;
}
```