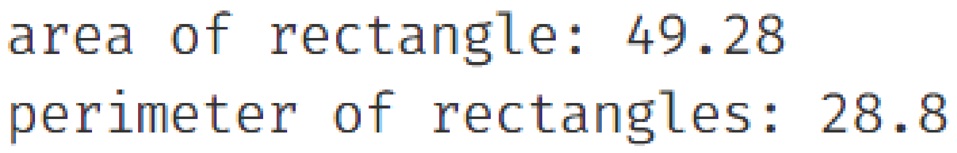
```cpp
/*1> Create a class representing a Rectangle with attributes for
length and width. Implement methods to calculate area and perimeter.*/
#include <iostream>
using namespace std;
class Rectangle
{
private:
    float length;
    float width;

public:
    void setter(float x, float y)
    {
        length = x, width = y;
    }
    float calculateArea()
    {
        return width * length;
    }
    float calculatePerimeter()
    {
        return 2 * (width + length);
    }
};
int main()
{
    Rectangle firstObj;
    firstObj.setter(5.6, 8.8);
    cout << "area of rectangle: " << firstObj.calculateArea() << endl;
    cout << "perimeter of rectangles: " << firstObj.calculatePerimeter() << endl;
    return 0;
}
```

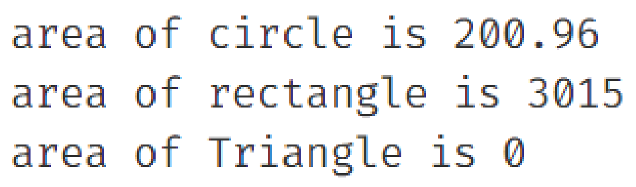
Output



```
area of rectangle: 49.28
perimeter of rectangles: 28.8
```

```cpp
/*2> Demonstrate function overloading by creating functions that calculate
     the area of different shapes (circle, triangle, rectangle).*/
#include<iostream>
using namespace std;
class shapes{
    public:
    float areaOfCircle(float r){
        return 3.14*r*r;
    }
    float areaOfTriangle(float w,float h){
        return (float)(1/2)*w*h;
    }

    float areaOfRectangle(float w,float h){
        return w*h;
    }
};
int main(){
    shapes obj1;
    cout<<"area of circle is "<<obj1.areaOfCircle(8)<<endl;
    cout<<"area of rectangle is "<<obj1.areaOfRectangle(45,67)<<endl;
    cout<<"area of Triangle is "<<obj1.areaOfTriangle(46,29.5)<<endl;
    return 0;
}
```

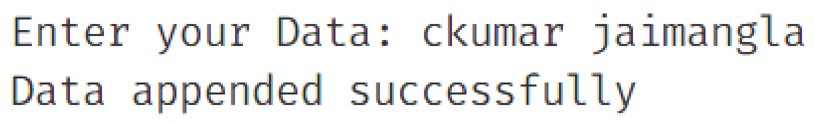
Output



```
area of circle is 200.96
area of rectangle is 3015
area of Triangle is 0
```

```cpp
/*3> Write a C++ program to open an output file 'a.txt' and append data to it.*/
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    ofstream fout("TestAppend.txt", ios::app);
    if (fout.is_open()){
        string temp;
        cout << "Enter your Data: ";
        getline(cin, temp);
        fout << temp;
        cout << "Data appended successfully" << endl;
    }
    fout.close();
    return 0;
}
```

Output



```
Enter your Data: ckumar jaimangla
Data appended successfully
```

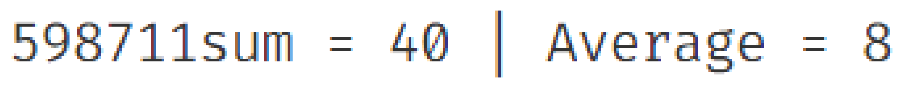
```cpp
/*4> Implement a class to represent a Bank Account with private
member variables and public methods for deposit and withdrawal.*/
#include <iostream>
using namespace std;
class PNB{
    float amount;
    string accountHolderName;
    string acNumber;
public:
    PNB(string name, string mob, float amount = 0){
        this->amount = amount;
        acNumber = mob;
        accountHolderName = name;
        cout << "Account open success!" << endl;
    }
    string deposit(float a, string number){
        if (number == acNumber){
            amount += a;
            return "deposit successfully";
        }
        else
            return "deposit failed";
    }
    string withdrawal(float a, string holder, string number){
        if (accountHolderName == holder && number == acNumber){
            amount -= a;
            return "withdrawal successfully";
        }
        else
            return "withdrawal failed! try again";
    }
    float getbalance(string accountNumber, string holder){
        if (accountNumber == acNumber && accountHolderName == holder)
            return amount;
        else
            return 0;
    }
};
int main(){
    PNB obj("chhotu", "8920785456");
    cout << obj.deposit(10000, "8920785456") << endl;
    cout << obj.withdrawal(5000, "chhotu", "8920785456") << endl;
    cout << obj.getbalance("8920785456", "chhotu") << endl;
}
```

Output

```
Account open success!
deposit successfully
withdrawal successfully
5000
```

```
/*5> Develop a program that dynamically allocates memory for an array of integers and
calculates their sum and average.*/
#include <iostream>
using namespace std;
int main(){
    int *ptrArr = new int[5];
    int sum = 0, average = 0,numberOfItem = 5;
    ptrArr[0] = 5;
    ptrArr[1] = 9;
    ptrArr[2] = 8;
    ptrArr[3] = 7;
    ptrArr[4] = 11;
    for (int i = 0; i < numberOfItem; i++){
        cout << ptrArr[i];
        sum += ptrArr[i];
    }
    average = sum / numberOfItem;
    cout << "sum = " << sum << " | Average = " << average << endl;
    return 0;
}
```

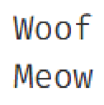
Output



```
598711sum = 40 | Average = 8
```

```
/*6> Create a base class "Animal" and derived classes "Cat" and "Dog" that override a
virtual method to produce different sounds.*/
#include <iostream>
using namespace std;
class Animal{
public:
    void virtual Sound() = 0;
};
class Cat : public Animal{
public:
    void Sound() { cout << "Meow" << endl; }
};
class Dog : public Animal{
public:
    void Sound() { cout << "Woof" << endl; }
};
int main(){
    Dog d;
    d.Sound();
    Cat c;
    c.Sound();
    return 0;
}
```

Output



```
Woof
Meow
```

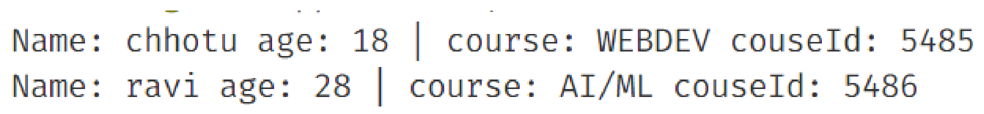
```
/*7> Write a C++ program that uses the new and delete operators to manage memory for
a student database.*/
#include <iostream>
using namespace std;
class Students
{
private:
    int roll;
    string name;
public:
    Students() {}
    Students(int roll, string name){
        this->roll = roll;
        this->name = name;
    }
    void displayStudentDetails(){
        cout << "Roll: " << roll << " | "
             << "Name: " << name << endl;
    }
    void setStudentDetails(int roll, string name){
        this->roll = roll;
        this->name = name;
    }
};
int main(){
    Students *ckumar = new Students(106, "chhotuKumar");
    ckumar->displayStudentDetails();

    Students *BCA = new Students[5];
    BCA[0].setStudentDetails(107, "Pankaj");
    BCA[1].setStudentDetails(108, "Abhishek");
    BCA[2].setStudentDetails(109, "gulshan");
    BCA[3].setStudentDetails(110, "zaid");
    BCA[4].setStudentDetails(111, "Mohit");

    for (int i = 0; i < 5; i++)
    {
        BCA[i].displayStudentDetails();
    }

    return 0;
}
```

Output

```
Roll: 106 | Name: chhotuKumar
Roll: 107 | Name: Pankaj
Roll: 108 | Name: Abhishek
Roll: 109 | Name: gulshan
Roll: 110 | Name: zaid
Roll: 111 | Name: Mohit
```

```
/*8> Implement a program using multiple inheritance to model a "Student" class
inheriting from both "Person" and "Course" classes.*/
#include <iostream>
using namespace std;
class Person{
protected:
    string name;
    int age;
public:
    Person(string name, int age){
        this->age = age;
        this->name = name;
    }
};
class Courses{
protected:
    string courseName;
    int courseId;
public:
    Courses(string courseName, int courseId){
        this->courseName = courseName;
        this->courseId = courseId;
    }
};
class Students : public Person, public Courses{
public:
    Students(string name, int age, string courseName, int courseId) : Person(name,
age), Courses(courseName, courseId) {}
    void DisplayStudentRecord(){
        cout << "Name: " << name << " age: " << age << " | course: " << courseName <<
" couseId: " << courseId << endl;
    }
};
int main(){
    Students s1("chhotu", 18, "WEBDEV", 5485);
    Students s2("ravi", 28, "AI/ML", 5486);
    s1.DisplayStudentRecord();
    s2.DisplayStudentRecord();
    return 0;
}
```

Output



```
Name: chhotu age: 18 | course: WEBDEV couseId: 5485
Name: ravi age: 28 | course: AI/ML couseId: 5486
```

```
/*9> Create a template function to find the maximum value from an array of different
data types (int, double, etc.).*/
#include <iostream>
using namespace std;
template <typename T>
T findMax(const T arr[], int size){
    if (size <= 0)
        throw invalid_argument("Array size must be greater than 0.");
    T maxVal = arr[0];
    for (int i = 1; i < size; ++i){
        if (arr[i] > maxVal)
            maxVal = arr[i];
    }
    return maxVal;
}
int main(){
    int intArray[] = {1, 5, 3, 8, 2};
    double doubleArray[] = {2.5, 1.1, 5.7, 3.3};
    int intMax = findMax(intArray, sizeof(intArray) / sizeof(intArray[0]));
    double doubleMax = findMax(doubleArray, sizeof(doubleArray) /
sizeof(doubleArray[0]));
    cout << "Maximum value in intArray: " << intMax << endl;
    cout << "Maximum value in doubleArray: " << doubleMax << endl;
    return 0;
}
```

Output

```
Maximum value in intArray: 8
Maximum value in doubleArray: 5.7
```

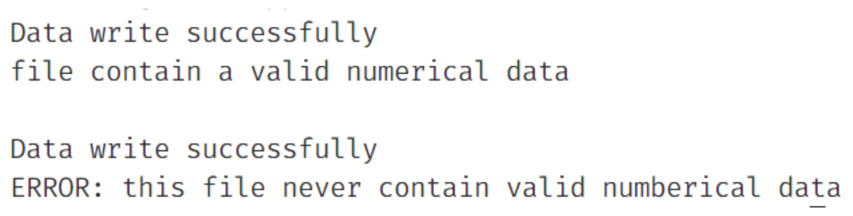
```
/*10> Write a C++ program to read a text file and count the number of characters in
it.*/
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    ifstream fin("temp.txt");
    char temp;
    int count = 0;
    if (!fin)
        cout << "File not able to open or file is not exist...!" << endl;
    else{
        while (!fin.eof()){
            count++;
            fin >> temp;
        }
        cout << "\ntotal number of character in a file is " << count << endl;
    }
    return 0;
}
```

Output

```
total number of character in a file is 30
```

```cpp
/*11> Implement a file handling program that reads numeric data from a file and
handles exceptions for incorrect data.*/
#include <iostream>
#include <fstream>
using namespace std;
class Files{
public:
    void Write(string Data){
        ofstream fw("temp.txt");
        if (fw.is_open()){
            fw << Data;
            cout << "Data write successfully" << endl;
            fw.close();
        }
    }
    string Read(){
        string temp;
        ifstream fr("temp.txt");
        if (fr.is_open()){
            try{
                getline(fr, temp);
                for (int i = 0; temp[i]; i++){
                    if (temp[i] >= '9' or temp[i] <= '0'){
                        throw "ERROR: this file never contain valid numberical data";
                        break;
                    }
                }
                return "file contain a valid numerical data";
            }
            catch (char const *e){
                return e;
            }
            fr.close();
        }
    }
};
int main(){
    Files file;
    file.Write("12345678");
    cout << file.Read() << endl<< endl;
    file.Write("chhotu kumar");
    cout << file.Read() << endl;
    return 0;
}
```

Output



```
Data write successfully
file contain a valid numerical data

Data write successfully
ERROR: this file never contain valid numberical data
```

```
/*12> Build a program that handles exceptions when dividing two numbers and outputs a
user-friendly error message.*/
#include <iostream>
#include <stdexcept>
using namespace std;
class Division{
    public:
    Division(int Nom, int Denom){
        try{
            if (Denom == 0)
                throw "Error: Division by zero is not allowed.";
            else
                cout << "Nom/Denom: " << Nom / Denom << endl;
        }
        catch (char const *e){
            cout << e;
        }
    }
};
int main(){
    Division div1(8, 4);
    Division div2(8, 0);
}
```

```
Nom/Denom: 2
Error: Division by zero is not allowed.
```

```
/*12> Write a C++ program to read data from a text file, sort it, and write the
sorted data back to the file.*/
#include <iostream>
#include <fstream>
#include <algorithm>
using namespace std;
int main(){
    string Data = "";
    ifstream fr("temp.txt");
    if (fr.is_open()){
        string temp;
        cout << "File opened successfully" << endl;
        do{
            getline(fr, temp);
            Data += " " + temp;
        } while (!fr.eof());
        fr.clear();
        sort(Data.begin(), Data.end());
        ofstream fw("temp.txt");
        if (fw.is_open()){
            fw << Data; cout << "Sorted data write successfully" << endl;
        }
    }
}
```

Output

```
File opened successfully
Sorted data write successfully
```

```
/*13>A file 'Employee.txt' contains empno and empname. Write a C++ program to add and
read contents of this file and search for an employee whose name is 'XYZ'.*/
#include <iostream>
#include <fstream>
using namespace std;
class Employee{
    int EmpNo;
    string EmpName;
    void Display(){
        cout << EmpNo << "  " << EmpName << endl;
    }
public:
    void InputRecord(){
        cout << "Enter Employament number:  ";
        cin >> EmpNo;
        fflush(stdin);
        cout << "Enter Employee Name     : ";
        getline(cin, EmpName);
        ofstream fout("Employee.txt", ios::app);
        if (!fout)
            cout << "Record are not saved successfully....";
        else
            fout << endl<< EmpNo << " " << EmpName;
    }
    void DisplayRecord(){
        ifstream fin("Employee.txt");
        if (!fin)
            cout << "File not able to open or read contant from this file location"
<< endl;
        else{
            do{
                fin >> EmpNo;
                fin >> EmpName;
                Display();
            } while (!fin.eof());
        }
    }
    void SearchRecord(){
        cout << "Enter your Name to search record:    ";
        string Name;
        getline(cin, Name);
        ifstream fin("Employee.txt");
        if (!fin)
            cout << "File not able to open or read contant from this file location"
<< endl;
        else{
            do{
                fin >> EmpNo;
                fin >> EmpName;
                if (EmpName.compare(Name) == 0){
                    Display();
                    return;
                }
            } while (!fin.eof());
            cout << "Record does not exist with " << Name;
        }}
};
int main(){
    Employee e1, e2, e3;
    e1.InputRecord();
    e2.InputRecord();
    e3.InputRecord();
    e1.DisplayRecord();
    e1.SearchRecord();
}
```

Output

```
Enter Employament number:  1
Enter Employee Name      : abc
Enter Employament number:  2
Enter Employee Name      : bcd
Enter Employament number:  3
Enter Employee Name      : cde
1  abc
2  bcd
3  cde
Enter your Name to search record:    bcd
2  bcd
```

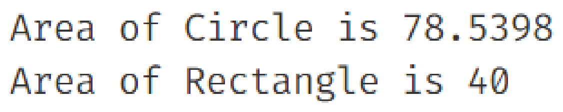
```
//14> Define a class Greatest and define instance member function to find Largest
among 3 numbers using classes.
#include <iostream>
using namespace std;
class LargestNumber{
    int a, b, c;
public:
    void input_number(){
        cout << "Enter three number: ";
        cin >> a >> b >> c;
    }
    void gestes_display(){
        if (a > b && a > c)
            cout << a << " is grater";
        else if (b > c)
            cout << b << " is grater";
        else
            cout << c << " is grater";
    }
};
int main(){
    LargestNumber a;
    a.input_number();
    a.gestes_display();
}
```

Output

```
Enter three number: 85456 48953 1452
85456 is grater
```

```
/*15> Design a base class "Shape" and derived classes for specific shapes. Implement
a virtual function to calculate their area.*/
#include <iostream>
#include <math.h>
using namespace std;
class Shape
{
public:
    virtual float caluculateArea() = 0;
};
class Circle : public Shape
{
    float radius;

public:
    Circle(float r) { radius = r; }
    float caluculateArea()
    {
        return M_PI * radius * radius;
    }
};
class Rectangle
{
protected:
    int length, breadth;

public:
    Rectangle(int l, int w)
    {
        length = l;
        breadth = w;
    }
    float calculateArea()
    {
        return length * breadth;
    }
};
int main(){
    Circle c(5);
    cout << "Area of Circle is " << c.caluculateArea() << endl;
    Rectangle r(5, 8);
    cout << "Area of Rectangle is " << r.calculateArea() << endl;
    return 0;
}
```

Output



```
Area of Circle is 78.5398
Area of Rectangle is 40
```

```
/*16> Write a C++ program to accept a password and throw an exception if the password
has less than 6 characters or does not contain a digit or does not contain any
special character or does not contain any capital letter.*/
#include <iostream>
using namespace std;
int main(){
    string password; int i;
    cout << "Enter your password: ";
    cin >> password;
    for (i = 0; password[i]; i++){
        if (!(password[i] >= 'a' and password[i] <= 'z')){
            i = -1;  break;
        }
    }
    try{
        if (password.length() < 6 or i != -1)
            throw "password is not strong";
        cout << "Login successfully" << endl;
    }
    catch (const char *msg){
        cout << msg << endl;
    }
}
```

Output

```
Enter your password: ck@123
Login successfully
```

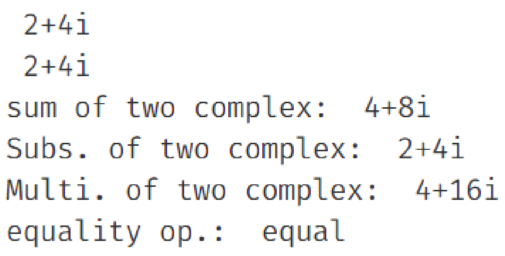
```
/*17> WAP that counts the total number of characters, words and lines inthe file.*/
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    string Str;
    int Characters = 0, Words = 0, Lines = 0, i;
    ifstream fin("temp.txt");
    do{
        getline(fin, Str);
        for (i = 0; Str[i]; i++)
            if (Str[i] == ' ')
                Words++;
        Characters += i;
        Words++;
        Lines++;
    } while (!fin.eof());
    cout << "Total number of characters are " << Characters << endl;
    cout << "Total number of lines are " << Lines << endl;
    cout << "Total number of words are " << Words << endl;
}
```

Output

```
Total number of characters are 14
Total number of lines are 1
Total number of words are 4
```

```
//18> Define a class Complex with appropriate instance variables and member
functions. Define following operators in the class: a. + b. - c. * d. ==
#include <iostream>
using namespace std;
class Complex{
    int r, i;
public:
    Complex(int, int);
    Complex() {}
    void showData(){
        cout << " " << r << "+" << i << "i" << endl;
    }
    Complex operator+(Complex x){
        Complex temp;
        temp.r = r + x.r;
        temp.i = i + x.i;
        return temp;
    }
    Complex operator-(Complex x){
        Complex temp;
        temp.r = r - x.r;
        temp.i = i - x.i;
        return temp;
    }
    Complex operator*(Complex x){
        Complex temp;
        temp.r = r * x.r;
        temp.i = i * x.i;
        return temp;
    }
    int operator==(Complex x){
        if (r == x.r && i == x.i)
            return 1;
    }
};
Complex::Complex(int x, int y){
    r = x,i = y;
}
int main(){
    Complex c1(2, 4), c2(2, 4);
    c1.showData();
    c2.showData();
    cout << "sum of two complex: ";
    Complex c3 = c2 + c1;
    c3.showData();
    cout << "Subs. of two complex: ";
    Complex c4 = c3 - c2;
    c4.showData();
    cout << "Multi. of two complex: ";
    Complex c5 = c2 * c1;
    c5.showData();
    cout << "equality op.:  ";
    int x = c1 == c2;
    if (x == 1)
        cout << "equal";
    else
        cout << "not equal";
}
```

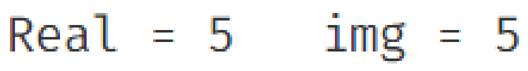
Output



```
 2+4i
 2+4i
sum of two complex:  4+8i
Subs. of two complex:  2+4i
Multi. of two complex:  4+16i
equality op.:  equal
```

```
/*19> .Write a C++ program to convert Primitive type to Complex type.
 Example -int main(){ Complex c1; Int x=5; c1=x; return 0; }*/
#include <iostream>
using namespace std;
class Complex
{
    int real, img;

public:
    Complex() {}
    Complex(int x)
    {
        real = x, img = x;
    }
    void Display()
    {
        cout << "Real = " << real << "   img = " << img << endl;
    }
};
int main()
{
    Complex c1;
    int x = 5;
    c1 = x;
    c1.Display();
    return 0;
}
```

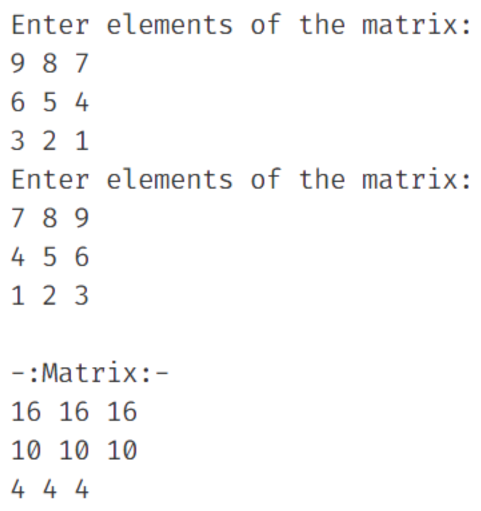
Output



```
Real = 5   img = 5
```

```
//20> Class Matrix {int a[3][3]; Public: //methods; };
// Let m1 and m2 are two matrices. Find out m3=m1+m2 (use operator overloading).
#include <iostream>
using namespace std;
class Matrix
{
   int a[3][3];
public:
    void Input_mat(){
        cout << "Enter elements of the matrix: \n";
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                cin >> a[i][j];
    }
    Matrix operator+(Matrix x){
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                x.a[i][j] = a[i][j] + x.a[i][j];
        return x;
    }
    void Show_mat(){
        cout << "\n-:Matrix:-\n";
        for (int i = 0; i < 3; i++){
            for (int j = 0; j < 3; j++)
                cout << a[i][j] << " ";
            cout << endl;
        }
    }
};
int main()
{
    Matrix m1, m2, m3;
    m1.Input_mat();
    m2.Input_mat();
    m3 = m2 + m1;
    m3.Show_mat();
    return 0;
}
```

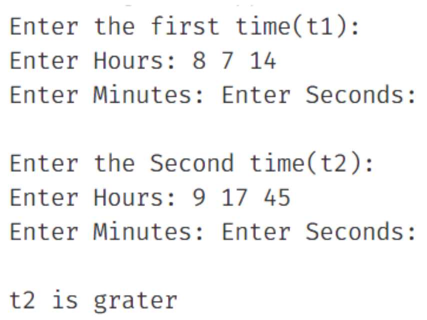
Output



```
Enter elements of the matrix:
9 8 7
6 5 4
3 2 1
Enter elements of the matrix:
7 8 9
4 5 6
1 2 3

-:Matrix:-
16 16 16
10 10 10
4 4 4
```

```
//21> Create a class Time which contains: - Hours - Minutes – Seconds, Write a C++
program using operator overloading for the following: 1. == : To check whether two
Times are the same or not. 2. >> : To accept the time. 3. << : To display the time.
#include <iostream>
using namespace std;
class Time{
    int HH, MM, SS;
public:
    Time(){
        cout << "Enter Hours: ";
        cin >> HH;
        cout << "Enter Minutes: ";
        cin >> MM;
        cout << "Enter Seconds: ";
        cin >> SS;
        cout << "\n\n";
    }
    int operator==(Time t){
        if (HH == t.HH && MM == t.MM && SS == t.SS)
            return 1;
        else
            return 0;
    }
    int operator>(Time t){
        if ((HH > t.HH) || (HH == t.HH && MM > t.MM) || (HH == t.HH && MM == t.MM &&
SS > t.SS))
            return 1;
        else
            return 0;
    }
    int operator<(Time t){
        if ((HH > t.HH) || (HH == t.HH && MM > t.MM) || (HH == t.HH && MM == t.MM &&
SS > t.SS))
            return 0;
        else
            return 1;
    }
};
int main(){
    cout << "Enter the first time(t1): " << endl;
    Time t1;
    cout << "Enter the Second time(t2): " << endl;
    Time t2;
    if (t1 == t2)
        cout << "Time are equal";
    else if (t1 > t2)
        cout << "t1 is grater";
    else if (t1 < t2)
        cout << "t2 is grater";
}
```

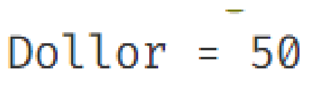
Output



```
Enter the first time(t1):
Enter Hours: 8 7 14
Enter Minutes: Enter Seconds:

Enter the Second time(t2):
Enter Hours: 9 17 45
Enter Minutes: Enter Seconds:

t2 is grater
```

```
/*22> Create a Dollar class and add necessary functions to support int to Dollar type
conversion. Example:-int main(){ int x = 50;Dollar d;d = x;d.display(); return 0;}*/
#include <iostream>
using namespace std;
class Dollar{
    int D;
public:
    void Display(){
        cout << "Dollor = " << D << endl;
    }
    Dollar(int x){
        D = x;
    }
    Dollar() {}
};
int main()
{
    int x = 50;
    Dollar d;
    d = x;
    d.Display();
    return 0;
}
```

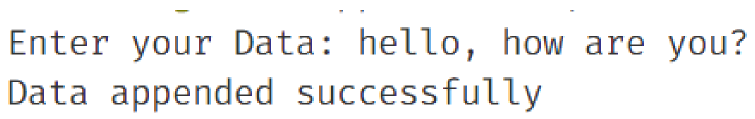
Output



```
Dollor = 50
```

```
/*23> Write a C++ program to open an output file 'a.txt' and append data to it.*/
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ofstream fout("TestAppend.txt", ios::app);
    if (fout.is_open())
{
        string temp;
        cout << "Enter your Data: ";
        getline(cin, temp);
        fout << temp;
        cout << "Data appended successfully" << endl;
    }
    fout.close();
    return 0;
}
```

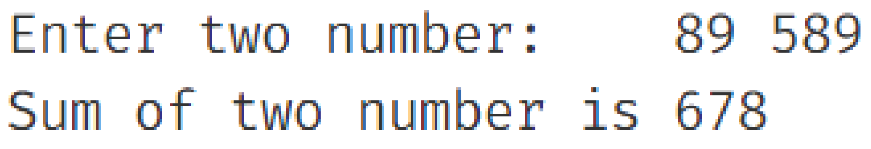
Output



```
Enter your Data: hello, how are you?
Data appended successfully
```

```
/24> *Write a C++ program to add two numbers using single inheritance. Accept these
two numbers from the user in base class and display the sum of these two numbers in
derived class.*/
#include <iostream>
using namespace std;
class BaseClass
{
protected:
    int x, y;

public:
    void InputData()
    {
        cout << "Enter two number:    ";
        cin >> x >> y;
    }
};
class DerivedClass : public BaseClass
{
public:
    int GetSumOfTwoNumbers()
    {
        return x + y;
    }
};
int main()
{

    DerivedClass d1;
    d1.InputData();
    cout << "Sum of two number is " << d1.GetSumOfTwoNumbers();
    return 0;
}
```

Output

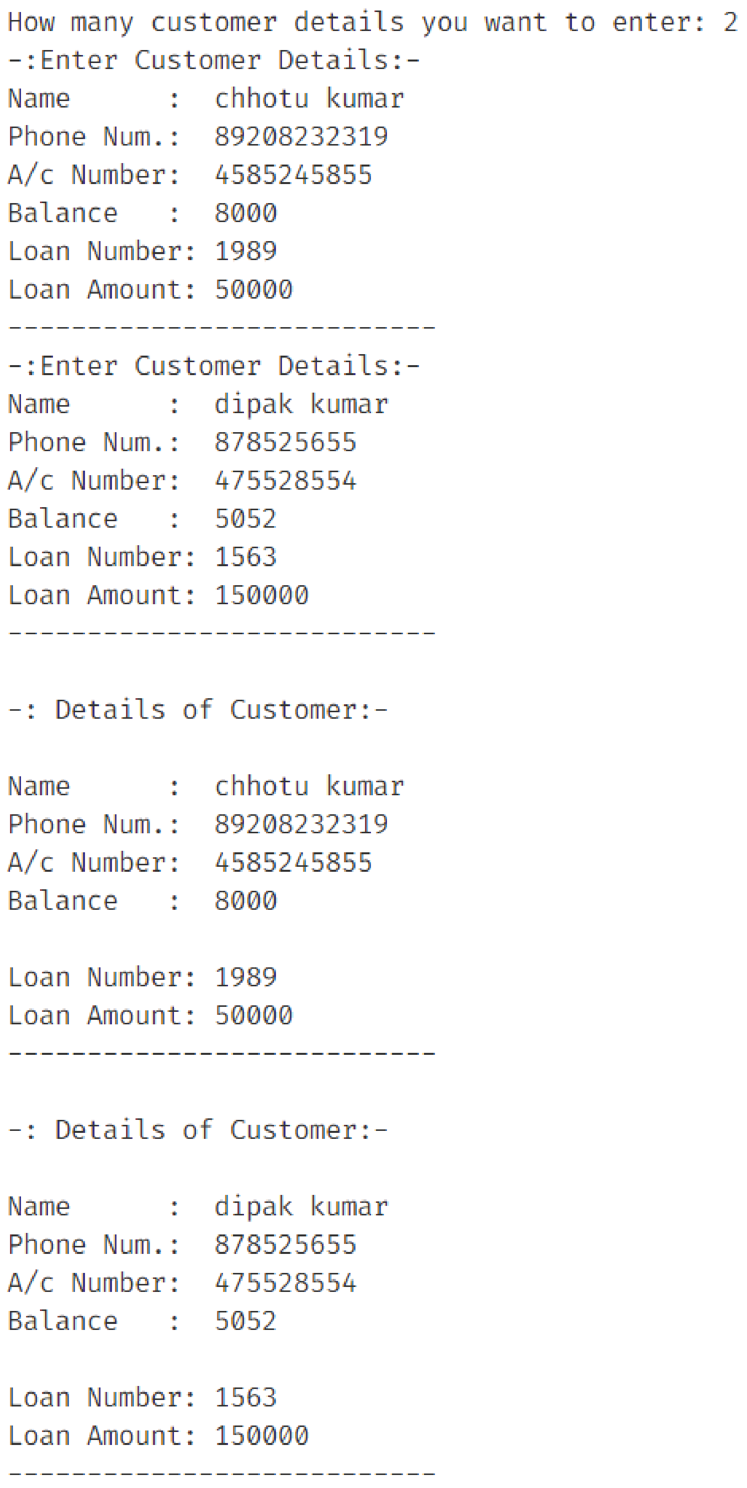


```
Enter two number:    89 589
Sum of two number is 678
```

```
/*25> In a bank, different customers have savings account. Some customers mayhave
taken a loan from the bank. So bank always maintain information aboutbank depositors
and borrowers.Design a Base class Customer (name, phone-number). Derive a
classDepositor(accno, balance) from Customer.Again, derive a class Borrower (loan-no,
loan-amt) from Depositor.Write necessary member functions to read and display the
details of ‘n’customers.*/

#include <iostream>
using namespace std;
class Custome{
protected:
    string Name, PhoneNumber;
};
class Depositor : protected Custome{
protected:
    string AccountNumber;
    int balance;
};
class Borrower : protected Depositor{
    int LoanNumber, LoanAmount;
public:
    void InputCustomerDetails(){
        fflush(stdin);
        cout << "-:Enter Customer Details:- " << endl;
        cout << "Name      :  ";
        getline(cin, Name);
        cout << "Phone Num.:  ";
        getline(cin, PhoneNumber);
        cout << "A/c Number:  ";
        getline(cin, AccountNumber);
        cout << "Balance   :  ";
        cin >> balance;
        cout << "Loan Number: ";
        cin >> LoanNumber;
        cout << "Loan Amount: ";
        cin >> LoanAmount;
        cout << "---------------------------" << endl;
    }
    void DisplayDetails(){
        cout << "\n-: Details of Customer:- \n"<< endl;
        cout << "Name      :  " << Name << endl;
        cout << "Phone Num.:  " << PhoneNumber << endl;
        cout << "A/c Number:  " << AccountNumber << endl;
        cout << "Balance   :  " << balance << endl;
        cout << "\nLoan Number: " << LoanNumber << endl;
        cout << "Loan Amount: " << LoanAmount << endl;
        cout << "---------------------------" << endl;
    }
};
```

```cpp
int main(){
    int N, i;
    cout << "How many customer details you want to enter: ";
    cin >> N;
    Borrower B[N];
    for (i = 0; i < N; i++)
        B[i].InputCustomerDetails();
    for (i = 0; i < N; i++)
        B[i].DisplayDetails();
    return 0;
}
```

Output



```
How many customer details you want to enter: 2
-:Enter Customer Details:-
Name       :  chhotu kumar
Phone Num.:  89208232319
A/c Number:  4585245855
Balance    :  8000
Loan Number: 1989
Loan Amount: 50000
--------------------------
-:Enter Customer Details:-
Name       :  dipak kumar
Phone Num.:  878525655
A/c Number:  475528554
Balance    :  5052
Loan Number: 1563
Loan Amount: 150000
--------------------------

-: Details of Customer:-

Name       :  chhotu kumar
Phone Num.:  89208232319
A/c Number:  4585245855
Balance    :  8000

Loan Number: 1989
Loan Amount: 50000
--------------------------

-: Details of Customer:-

Name       :  dipak kumar
Phone Num.:  878525655
A/c Number:  475528554
Balance    :  5052

Loan Number: 1563
Loan Amount: 150000
--------------------------
```