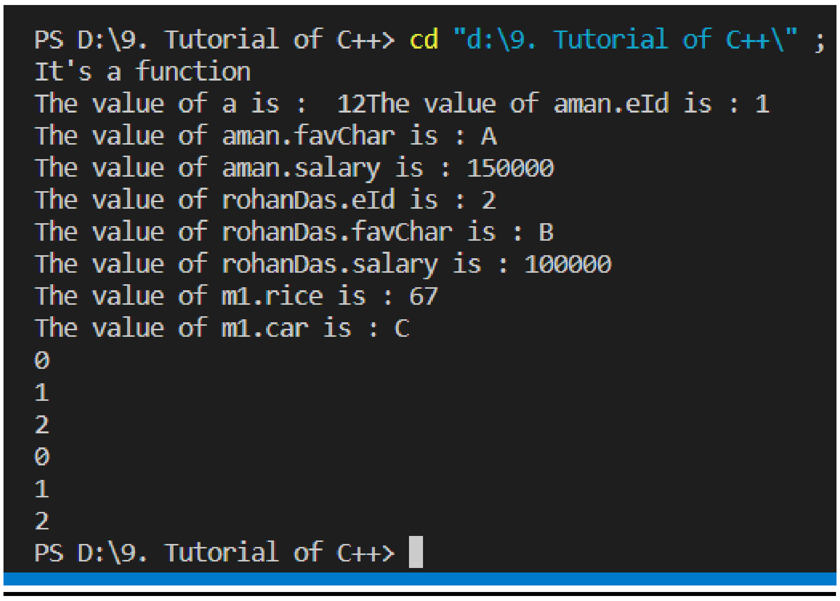
INDEX :

Structure, Union And Enum :

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

typedef struct employee{ //By using typedaf we can write 'employee' in short form like 'xy'
    int eId;
    char favChar;
    float salary;
    void func(){
        cout<<"It's a function"<<endl;
    }
    int getNum(int a){
        return a;
    }
}ep;  // Here we must write the short form of employee.

union money{
    int rice;
    char car;    // We can use only one at a time as memory is being shared.
    float pounds;
};

int main(){
    employee aman;
    ep rohanDas;

    aman.eId = 1;
    aman.favChar = 'A';
    aman.salary = 150000;
    aman.func();
    cout<<"The value of a is :  "<<aman.getNum(12);

    rohanDas.eId = 2;
    rohanDas.favChar = 'B';
    rohanDas.salary = 100000;

    cout<<"The value of aman.eId is : "<<aman.eId<<endl;

    cout<<"The value of aman.favChar is : "<<aman.favChar<<endl;

    cout<<"The value of aman.salary is : "<<aman.salary<<endl;

    cout<<"The value of rohanDas.eId is : "<<rohanDas.eId<<endl;

    cout<<"The value of rohanDas.favChar is : "<<rohanDas.favChar<<endl;

    cout<<"The value of rohanDas.salary is : "<<rohanDas.salary<<endl;


    union money m1;

    m1.rice = 34;

    m1.car = 'C';

    cout<<"The value of m1.rice is : "<<m1.rice<<endl; //This will give garbage value.

    cout<<"The value of m1.car is : "<<m1.car<<endl;


    enum Meal{breakfast, lunch, dinner};

    cout<<breakfast<<endl;

    cout<<lunch<<endl;

    cout<<dinner<<endl;


    // Now Meal become a data type, so we can use it as below.

    Meal n1 = breakfast;

    Meal n2 = lunch;

    Meal n3 = dinner;

    cout<<n1<<endl;

    cout<<n2<<endl;

    cout<<n3<<endl;


return 0;

}
```

Output :-

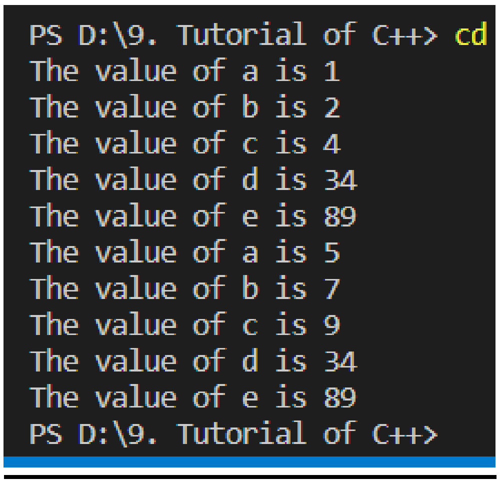


```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++\" ;
It's a function
The value of a is :  12The value of aman.eId is : 1
The value of aman.favChar is : A
The value of aman.salary is : 150000
The value of rohanDas.eId is : 2
The value of rohanDas.favChar is : B
The value of rohanDas.salary is : 100000
The value of m1.rice is : 67
The value of m1.car is : C
0
1
2
0
1
2
PS D:\9. Tutorial of C++>
```

Class :

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

class Employee{
    private:
    int a, b, c;
    public:
    int d, e;
    void setData(int a1, int b1, int c1);  // --> Declaration
    void getData(){
        cout<<"The value of a is "<<a<<endl;
        cout<<"The value of b is "<<b<<endl;
        cout<<"The value of c is "<<c<<endl;
        cout<<"The value of d is "<<d<<endl;
        cout<<"The value of e is "<<e<<endl;
    }
};

void Employee :: setData(int a1, int b1, int c1){
    a = a1;
    b = b1;
    c = c1;
}

int main(){
    Employee itsme;
    // itsme.a = 134;   --> This will throw error as 'a' is private member variable.
    itsme.d = 34;
    itsme.e = 89;
    itsme.setData(1, 2, 4);
    itsme.getData();
    itsme.setData(5, 7, 9);
    itsme.getData();
return 0;
}
```

Output :-



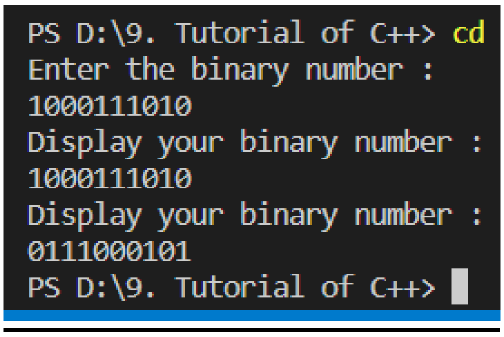
```
PS D:\9. Tutorial of C++> cd
The value of a is 1
The value of b is 2
The value of c is 4
The value of d is 34
The value of e is 89
The value of a is 5
The value of b is 7
The value of c is 9
The value of d is 34
The value of e is 89
PS D:\9. Tutorial of C++>
```

Nesting Of Member Variables :

```
#include<iostream>

using namespace std;

// OOPs - Classes and objects

// Class -->Extension of structure (in C)

class Binary{

    string s;

    void chk_bin();

    public:

    void read();

    void ones();

    void display();

};


void Binary :: read(){

    cout<<"Enter the binary number : "<<endl;

    cin>>s;

}


void Binary :: chk_bin(){

    for(int i=0; i<s.length(); i++){

        if(s.at(i) != '0' && s.at(i) != '1'){

            cout<<"Incorrect binary format "<<endl;

            exit(0);

        }

    }

}


void Binary :: ones(){

    chk_bin();  // This is nesting of the function

    for(int i=0; i<s.length(); i++){

        if(s.at(i)=='0'){

            s.at(i)='1';

        }

        else
```

```cpp
        s.at(i)='0';
    }
}

void Binary :: display(){
    cout<<"Display your binary number :\n"<<s;
    cout<<endl;
}

int main(){
    Binary b;
    b.read();
    // b.chk_bin();    // -->This will throw error as this is private member function
    b.display();
    b.ones();
    b.display();
return 0;
}
```

Output :-

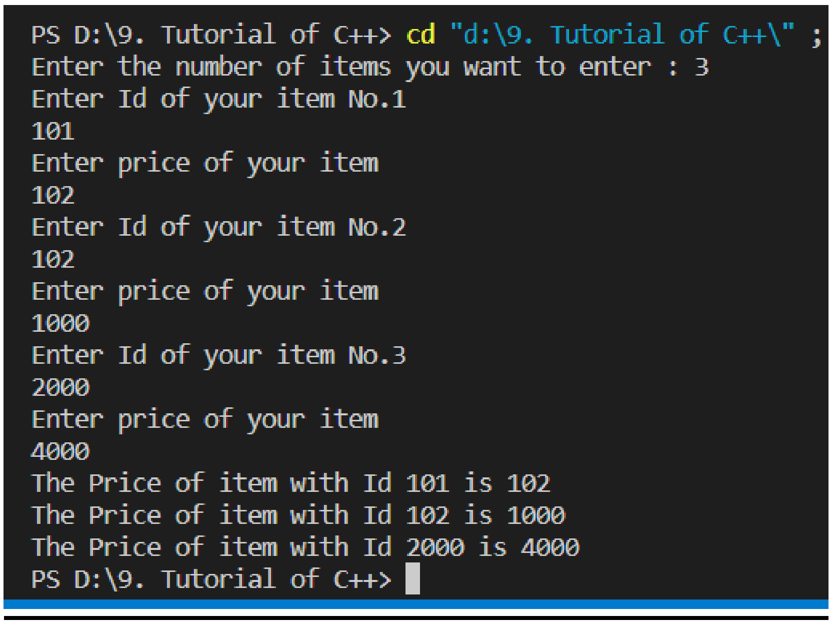


```
PS D:\9. Tutorial of C++> cd
Enter the binary number :
1000111010
Display your binary number :
1000111010
Display your binary number :
0111000101
PS D:\9. Tutorial of C++>
```

Oject Memory Allocation :

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

class Shop{
    int itemId[100];
    int itemPrice[100];
    int counter;
    public:
    void initCounter(){ counter = 0; }
    void setPrice();
    void displayPrice();
};

void Shop :: setPrice(){
    cout<<"Enter Id of your item No."<<counter+1<<endl;
    cin>>itemId[counter];
    cout<<"Enter price of your item"<<endl;
    cin>>itemPrice[counter];
    counter++;
}

void Shop :: displayPrice(){
    for(int i=0; i<counter; i++){
        cout<<"The Price of item with Id "<<itemId[i]<<" is "<<itemPrice[i]<<endl;
    }
}

int main(){
    int n;
    cout<<"Enter the number of items you want to enter : ";
    cin>>n;
    Shop dukan;
    dukan.initCounter();
    for(int i=0; i<n; i++){
        dukan.setPrice();

    }

    dukan.displayPrice();


return 0;

}
```

Output :-

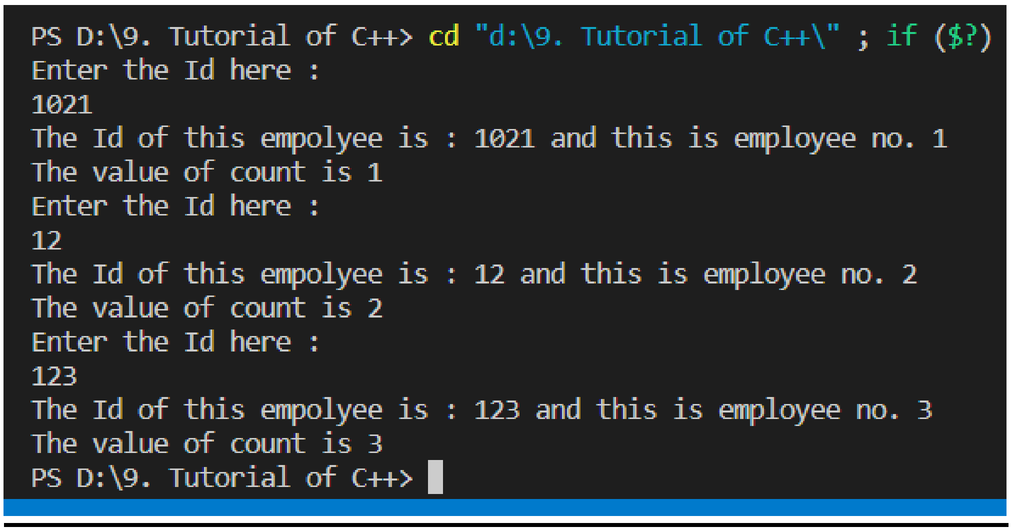


```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++\" ;
Enter the number of items you want to enter : 3
Enter Id of your item No.1
101
Enter price of your item
102
Enter Id of your item No.2
102
Enter price of your item
1000
Enter Id of your item No.3
2000
Enter price of your item
4000
The Price of item with Id 101 is 102
The Price of item with Id 102 is 1000
The Price of item with Id 2000 is 4000
PS D:\9. Tutorial of C++>
```

Static Data Members :

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

class Employee{
    static int count;  //By default its value is start with 0 and also know as class variable.
    int Id;
    public:
    void setData(){
        cout<<"Enter the Id here : "<<endl;
        cin>>Id;
        count++;
    }
    void getData(){
        cout<<"The Id of this empolyee is : "<<Id<<" and this is employee no. "<<count<<endl;
    }

    static void getCount(){
        // Static function will excess only static variable.
        // cout<<Id;  -->This will throw error.
        cout<<"The value of count is "<<count<<endl;
    }
};

// 'count' is the static data members of class Employee and Its neccessary to show it
int Employee :: count;  // at here and we can initialise 'count' from here only.

int main(){
    Employee harry, aman, lovish;
    harry.setData();
    harry.getData();
    Employee::getCount();

    aman.setData();
    aman.getData();
    Employee::getCount();


    lovish.setData();

    lovish.getData();

    Employee::getCount();


return 0;

}
```

Output :-

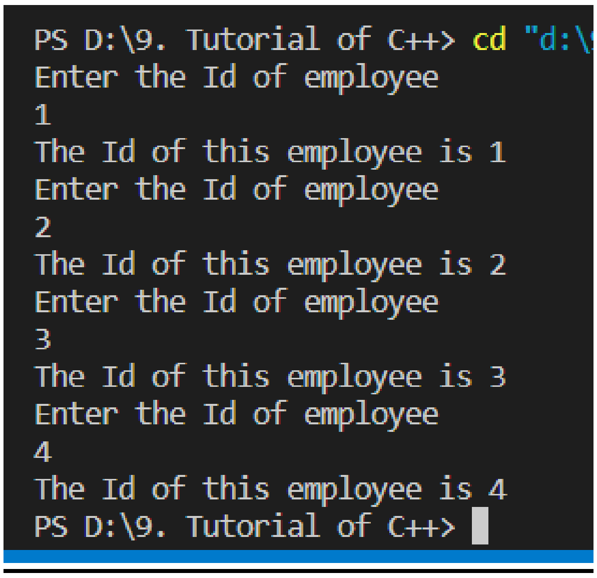


```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++\" ; if ($?)
Enter the Id here :
1021
The Id of this empolyee is : 1021 and this is employee no. 1
The value of count is 1
Enter the Id here :
12
The Id of this empolyee is : 12 and this is employee no. 2
The value of count is 2
Enter the Id here :
123
The Id of this empolyee is : 123 and this is employee no. 3
The value of count is 3
PS D:\9. Tutorial of C++>
```

Array Of Objects :

```cpp
#include<iostream>

using namespace std;


class Employee{

    int Id;

    int salary;

    public:

    void setId(){

        salary = 122;

        cout<<"Enter the Id of employee"<<endl;

        cin>>Id;

    }


    void getId(){

        cout<<"The Id of this employee is "<<Id<<endl;

    }

};


int main(){

    // Employee Aman, Nikki, harry, rohan;

    // harry.setId();

    // harry.getId();

    // Here we use array for object.

    Employee fb[4];

    for(int i=0; i<4; i++){

        fb[i].setId();

        fb[i].getId();

    }


return 0;

}
```

Output :-

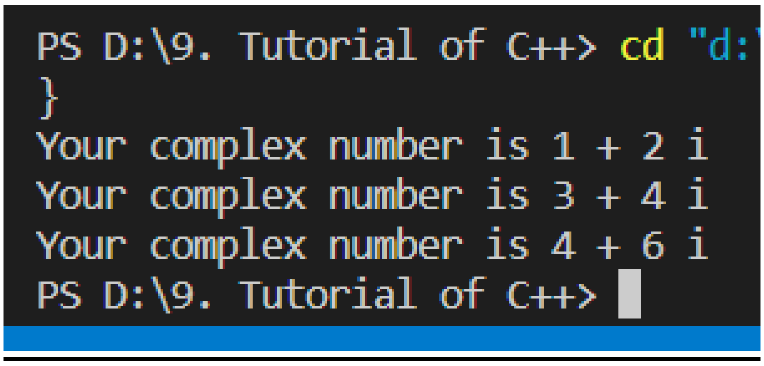


```
PS D:\9. Tutorial of C++> cd "d:\
Enter the Id of employee
1
The Id of this employee is 1
Enter the Id of employee
2
The Id of this employee is 2
Enter the Id of employee
3
The Id of this employee is 3
Enter the Id of employee
4
The Id of this employee is 4
PS D:\9. Tutorial of C++>
```

Passing Objects As A Function :

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

class Complex{
    int a, b;
    public:
    void setData(int v1, int v2){
        a = v1;
        b = v2;
    }

    void setDataBySum(Complex o1, Complex o2){
        a = o1.a + o2.a;
        b = o1.b + o2.b;
    }

    void print(){
         cout<<"Your complex number is "<<a<<" + "<<b<<" i"<<endl;
    }
};

int main(){
    Complex c1, c2, c3;
    c1.setData(1, 2);
    c1.print();

    c2.setData(3, 4);
    c2.print();

    c3.setDataBySum(c1, c2);
    c3.print();

return 0;
}
```

Output :-

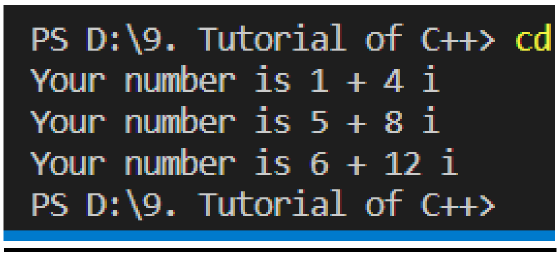


```
PS D:\9. Tutorial of C++> cd "d:
}
Your complex number is 1 + 2 i
Your complex number is 3 + 4 i
Your complex number is 4 + 6 i
PS D:\9. Tutorial of C++>
```

Friend Function :

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

class Complex{
    int a, b;
    public:
    void setNumber(int n1, int n2){
        a = n1;
        b = n2;
    }

    void printNumber(){
        cout<<"Your number is "<<a<<" + "<<b<<" i"<<endl;
    }
// Below line means that non member - sumcomplex function is allowed to do anything
// with my private parts(data).
    friend Complex sumComplex(Complex o1, Complex o2);
};

Complex sumComplex(Complex o1, Complex o2){
    Complex o3;
    o3.setNumber((o1.a + o2.a), (o1.b + o2.b));
    return o3;
}

int main(){
    Complex c1, c2, sum;
    c1.setNumber(1, 4);
    c1.printNumber();

    c2.setNumber(5, 8);
    c2.printNumber();

    sum = sumComplex(c1, c2);
    sum.printNumber();


return 0;

}


/* Properties of friend functions.

1. Not in the scope of the class.

2. Since it is not in the scope of the class, it cannot be called

from the object of that class such that --> c1.sumComplex() == Invalid

3. Can be invoked without the help of any objects.

4. Usually contains the objects as arguments.

5. Can be declared inside public or private section of the class.

6. It cannot access the members directly by thier names and need object_name.member_name

to acess any member.

*/
```

<u>Output :-</u>



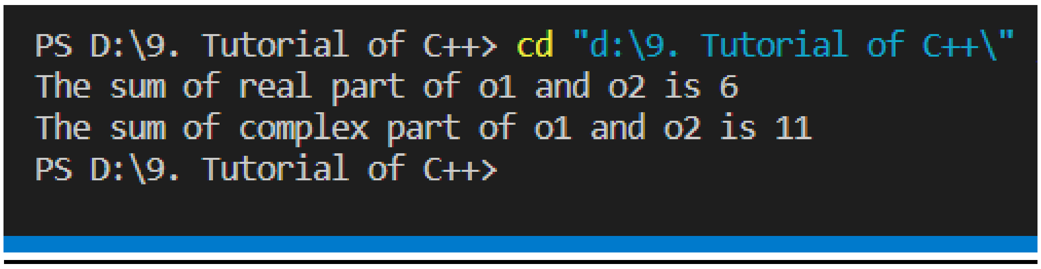
```
PS D:\9. Tutorial of C++> cd
Your number is 1 + 4 i
Your number is 5 + 8 i
Your number is 6 + 12 i
PS D:\9. Tutorial of C++>
```

Friend Class **:**

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

// Forward declaration
class Complex;

class Calculator{
    public:
    int add(int a, int b){
        return (a+b);
    }

    int sumRealComplex(Complex, Complex);
    int sumCompComplex(Complex, Complex);
};

class Complex{
    int a, b;
    // Individually declaring functions as friends
    /* friend int Calculator :: sumRealComplex(Complex, Complex);
    friend int Calculator :: sumCompComplex(Complex, Complex); */

    // Aliter: Declaring the entire calculator class as friend
    friend class Calculator;
    public:
    void setNumber(int n1, int n2){
        a = n1;
        b = n2;
    }

};

int Calculator :: sumRealComplex(Complex o1, Complex o2){
    return (o1.a + o2.a);
```

```cpp
}
int Calculator :: sumCompComplex(Complex o1, Complex o2){
    return (o1.b + o2.b);
}

int main(){
    Complex o1, o2;
    o1.setNumber(1, 4);
    o2.setNumber(5, 7);
    Calculator calc;
    cout<<"The sum of real part of o1 and o2 is "<<calc.sumRealComplex(o1, o2)<<endl;
    cout<<"The sum of complex part of o1 and o2 is "<<calc.sumCompComplex(o1, o2)<<endl;

return 0;
}
```

Output :-

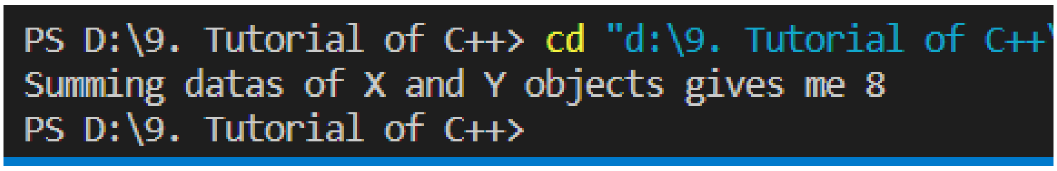


```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++\"
The sum of real part of o1 and o2 is 6
The sum of complex part of o1 and o2 is 11
PS D:\9. Tutorial of C++>
```

More On Friend Class :

```cpp
#include<iostream>
using namespace std;
class Y;
class X{
    int data;
    public:
    void setValue(int value){
        data = value;
    }
    friend void add(X, Y);
};
class Y{
    int num;
    public:
    void setValue(int value){
        num = value;
    }
    friend void add(X, Y);
};

void add(X o1, Y o2){
    cout<<"Summing datas of X and Y objects gives me "<<o1.data + o2.num;
}

int main(){
    X a;
    a.setValue(3);

    Y b;
    b.setValue(5);

    add(a, b);
return 0;
}
```

Output :-

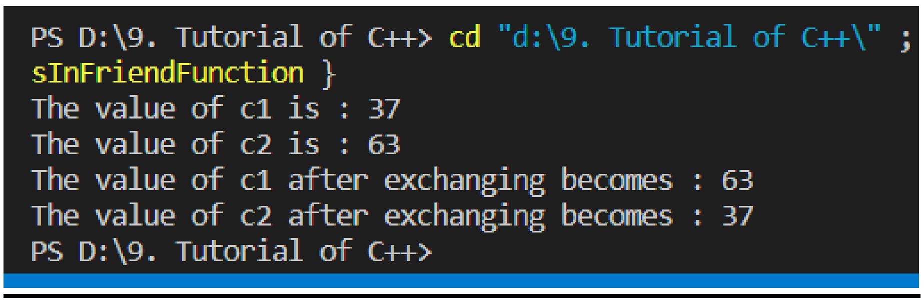


```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++
Summing datas of X and Y objects gives me 8
PS D:\9. Tutorial of C++>
```

Swaping Numbers by Friend Function :

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

class c2;
class c1{
    int val1;
    friend void exchange(c1 &, c2 &);
    public:
    void indata(int a){
        val1 = a;
    }

    void display(){
        cout<<val1<<endl;
    }
};

class c2{
    int val2;
    friend void exchange(c1 &, c2 &);
    public:
    void indata(int a){
        val2 = a;
    }

    void display(){
        cout<<val2<<endl;
    }
};

void exchange(c1 &x, c2 &y){
    x.val1 = x.val1 + y.val2;
    y.val2 = x.val1 - y.val2;
    x.val1 = x.val1 - y.val2;
}


int main(){
    c1 oc1;
    c2 oc2;

    oc1.indata(37);
    oc2.indata(63);
    cout<<"The value of c1 is : ";
    oc1.display();
    cout<<"The value of c2 is : ";
    oc2.display();

    exchange(oc1, oc2);
    cout<<"The value of c1 after exchanging becomes : ";
    oc1.display();
    cout<<"The value of c2 after exchanging becomes : ";
    oc2.display();

return 0;
}
```

Output :-



```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++\" ;
sInFriendFunction }
The value of c1 is : 37
The value of c2 is : 63
The value of c1 after exchanging becomes : 63
The value of c2 after exchanging becomes : 37
PS D:\9. Tutorial of C++>
```