```cpp
// -------- Copy Constructor : shallow copy without ptrs--------------------

#include <iostream>
using namespace std;

class account{
    private:
        string customer_name {};
        int balance {};
        int cibil {};
    public:
        void deposit(int amount);
        void withdraw(int amount);
        void getbalance();

        account(string name = "Happy" , int val = 600,  int bal = 0);
        ~account() {
            cout << "Destructor called for object " << customer_name << endl;
        }

        account(const account &source):
customer_name{"Hello"},cibil{source.cibil}, balance{source.balance}{
            cout << "Copy constructor " << customer_name << endl;
        }
};

account::account (string name, int val, int bal):
customer_name{name},cibil{val}, balance{bal} {
    cout << "constructor called for object " << customer_name << endl;
}

void account::deposit(int amount) {
    balance += amount;
}
void account::withdraw(int amount) {
    balance -= amount;
}
void account::getbalance(){
    cout << customer_name << " having balance: " << balance << " & cibil val.:
" << cibil << endl;
}

int main(){
    account savings_acc3("cust2",500,1500);
    savings_acc3.deposit(1500);
    savings_acc3.withdraw(1000);
    savings_acc3.getbalance();
    account new_acc{savings_acc3};
    new_acc.deposit(500);
    new_acc.getbalance();

    return 0;
}
```

Result:

```
constructor called for object cust2

cust2 having balance: 2000 & cibil val.: 500

Copy constructor Hello

Hello having balance: 2500 & cibil val.: 500

Destructor called for object Hello

Destructor called for object cust2
```

```cpp
//----copy constructor - shallow copy with ptrs---------

#include <iostream>
using namespace std;

class account{
    private:
        int *balance;
    public:
        void set_bal(int amount);
        int getbalance();

        account(int bal = 100);
        ~account() {
            delete balance;
            cout << "Destructor called for object " << endl;
        }

        account(const account &source):balance{source.balance}{
            cout << "Copy constructor - shallow copy " << endl;
        }
};

account::account (int bal) {
    cout << "Constructor created" << endl;
    balance = new int;
    *balance = bal;
}
void account::set_bal(int amount) {
    *balance = amount;
}
int account::getbalance(){
    return *balance;
}

int main()
{
    account savings_acc3{500};
    account new_acc{savings_acc3};
    cout << new_acc.getbalance() << endl;

    return 0;
}
```

Result:

```
Constructor created

Copy constructor - shallow copy

500

Destructor called for object

free(): double free detected in tcache 2
```

```cpp
// ----------------copy constructor – deep copy with ptrs-------------

#include <iostream>
using namespace std;

class account{
    private:
        int *balance;
    public:
        void set_bal(int amount);
        int getbalance();

        account(int bal = 100);
        ~account() {
            delete balance;
            cout << "Destructor called for object- freeing data "  << balance
<< endl;
        }

        account(const account &source):account{*source.balance}{
            cout << "Copy constructor - Deep copy " << endl;
        }

};

account::account (int bal) {
    cout << "Constructor created" << endl;
    balance = new int;
    *balance = bal;
}

void account::set_bal(int amount) {
    *balance = amount;
}
int account::getbalance(){
    return *balance;
}
```

```cpp
int main(){
    account savings_acc3{500};

    account new_acc{savings_acc3};
    cout << new_acc.getbalance() << endl;

    return 0;
}
```

Result:

```
Constructor created

Constructor created

Copy constructor - Deep copy

500

Destructor called for object- freeing data 0x560476eef2e0

Destructor called for object- freeing data 0x560476eef2c0
```

```cpp
// ------------this pointer------------

#include <iostream>
using namespace std;

class account{
    private:
        string customer_name {};
        int balance {};
        int cibil {};
    public:
        void setbalance(int balance);
        void deposit(int amount);
        void withdraw(int amount);
        void getbalance();

        account(string name = "Happy" , int val = 600,  int bal = 0);
        ~account() {
            cout << "Destructor called for object " << customer_name << endl;

        }
};

account::account (string name, int val, int bal):
customer_name{name},cibil{val},balance{ bal}{
    cout << "constructor called for object " << customer_name << endl;   }

void account::setbalance(int balance) {
    this->balance = balance;
};
void account::deposit(int amount) {
    balance += amount;
}
void account::withdraw(int amount) {
    balance -= amount;
}
void account::getbalance(){
    cout << customer_name << " having balance: " << balance << " & cibil val.:
" << cibil << endl;   }

int main(){
    account savings_acc3("cust2",500,1500);
    savings_acc3.deposit(1500);
    savings_acc3.withdraw(1000);
    savings_acc3.getbalance();

    account new_acc{savings_acc3};
    new_acc.deposit(500);
    new_acc.getbalance();

    return 0;
}
```

Result:

```
constructor called for object cust2

cust2 having rs.: 2000 & cibil val.: 500

cust2 having rs.: 2500 & cibil val.: 500

Destructor called for object cust2

Destructor called for object cust2
```

```cpp
// ---------- static members of a class -------------------

#include <iostream>
using namespace std;

class account{
    private:
        static int customer_count;
        string customer_name {};
        int balance {};
        int cibil {};
    public:
        void getbalance();
        static int get_customer_count();

        account(string name = "Happy" , int val = 600,  int bal = 0);
        ~account() {
            cout << "Destructor called for object " << customer_name << endl;
        }
};

int account::customer_count {0};

int account::get_customer_count(){
    return customer_count;
}

account::account (string name, int val, int bal):
customer_name{name},cibil{val},balance{ bal}{
    customer_count++;
    cout << "constructor called for object " << customer_name << endl;
}

void account::getbalance(){
    cout << customer_name << " having rs.: " << balance << " & cibil val.: "
<< cibil << endl;
}



int main(){

    cout << "no. of customers in bank are " << account::get_customer_count()
<< endl;
    account savings_acc1;
    savings_acc1.getbalance();
    cout << "no. of customers in bank are " << account::get_customer_count()
<< endl;
    account savings_acc2("cust1");
    cout << "no. of customers in bank are " << account::get_customer_count()
<< endl;
    account savings_acc3("cust2",500,1500);
    cout << "no. of customers in bank are " << account::get_customer_count()
<< endl;
    account new_acc{savings_acc3};  //
    cout << "after copy constructor no. of customers in bank are " <<
account::get_customer_count() << endl;

    return 0;
}
```

Result:

```
no. of customers in bank are 0

constructor called for object Happy

Happy having rs.: 0 & cibil val.: 600

no. of customers in bank are 1

constructor called for object cust1

no. of customers in bank are 2

constructor called for object cust2

no. of customers in bank are 3

after copy constructor no. of customers in bank are 3

Destructor called for object cust2

Destructor called for object cust2

Destructor called for object cust1

Destructor called for object Happy
```

```cpp
//-----------friend of a class --------------

#include <iostream>
using namespace std;

class account{
    private:
        string customer_name {};
        int balance {};
        int cibil {};
    public:
    friend void getdetails(account acc);

        account(string name = "Happy" , int val = 600,  int bal = 0);
        ~account() {
            cout << "Destructor called for object " << customer_name << endl;
        }
};

account::account (string name, int val, int bal):
customer_name{name},cibil{val},balance{ bal}{
    cout << "constructor called for object " << customer_name << endl;
}

void getdetails(account acc){
    cout << acc.customer_name << " having rs.: " << acc.balance << " & cibil
val.: " << acc.cibil << endl;
}

int main(){
    account savings_acc3("cust2",500,1500);
    getdetails(savings_acc3);
    return 0;
}
```

Result:

```
constructor called for object cust2

cust2 having rs.: 1500 & cibil val.: 500

Destructor called for object cust2

Destructor called for object cust2
```