```
// ----operator overloading : + operator as member function------

#include <iostream>
using namespace std;

class account{
    private:
        int balance {};
        int cibil {};
    public:
        void getdetails();

        account operator+(int value);

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

account account::operator+(int value){
    cout<< "in operator bal: " << balance << endl;
    cout<< "in operator val: " << value << endl;
    return account(900,balance+ value);  // 900 for the cibil score
}

account::account ( int val, int bal): cibil{val},balance{ bal}{
    cout << "constructor called for balance " << balance <<" & value " << cibil <<
endl;
}

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

int main(){
    account savings_acc1(500,1500);
    savings_acc1.getdetails();
    account savings_acc2 = savings_acc1+300;
    savings_acc2.getdetails();
    return 0;
}
```

result:

```
constructor called for balance 1500 & value 500
 having rs.: 1500 & cibil val.: 500
in operator bal: 1500
in operator val: 300
constructor called for balance 1800 & value 900
 having rs.: 1800 & cibil val.: 900
Destructor called for object
Destructor called for object
```

```cpp
// -----operator overloading global function with both operand as constant object--

#include <iostream>
using namespace std;

class account{
    private:
        int balance {};
    public:
        void getdetails();

        friend account operator+( const account &op1, const account &op2);

        account( int bal = 0);
        ~account() {
            cout << "Destructor called for object " << endl;
        }
};

account operator+(const account &op1, const account &op2){
    cout<< "in operator bal: " << op1.balance  << endl;
    return account(op1.balance+ op2.balance);  // 900 for the cibil score
}

account::account ( int bal): balance{ bal}{
    cout << "constructor called for balance " << balance << endl;
}

void account::getdetails(){
    cout<< " having rs.: " << balance  << endl;
}

int main(){
    account savings_acc1(1500);
    account savings_acc2(1000);
    account savings_acc4 = savings_acc1+savings_acc2;
    savings_acc4.getdetails();
    return 0;
}
```

Result:

```
constructor called for balance 1500
constructor called for balance 1000
in operator bal: 1500
constructor called for balance 2500
 having rs.: 2500
Destructor called for object
Destructor called for object
Destructor called for object
```

```cpp
//-----operator overloading global function with one operand as object----
#include <iostream>
using namespace std;

class account{
    private:
        int balance {};
    public:
        void getdetails();

        friend account operator+( account op1, int op2);

        account( int bal = 0);
        ~account() { cout << "Destructor called for object " << endl;}
};

account operator+(account op1, int op2){
    cout<< "in operator bal: " << op1.balance  << endl;
    return account(op1.balance+ op2);  // 900 for the cibil score
}

account::account ( int bal): balance{ bal}{
    cout << "constructor called for balance " << balance << endl;
}

void account::getdetails(){
    cout<< " having rs.: " << balance  << endl;
}

int main(){
    account savings_acc1(1500);
    account savings_acc3 = savings_acc1+300;
    savings_acc3.getdetails();
    return 0;
}
```

Result:

```
constructor called for balance 1500
in operator bal: 1500
constructor called for balance 1800
Destructor called for object
 having rs.: 1800
Destructor called for object
Destructor called for object
```

```cpp
//-------------operator overloading global function with one operand as
constant  object---------------
#include <iostream>
using namespace std;

class account{
    private:
        int balance {};
    public:
        void getdetails();

        friend account operator+(const account &op1, int op2);

        account( int bal = 0);
        ~account() { cout << "Destructor called for object " << endl;}
};

account operator+(const account &op1, int op2){
    cout<< "in operator bal: " << op1.balance  << endl;
    return account(op1.balance+ op2);  // 900 for the cibil score
}

account::account ( int bal): balance{ bal}{
    cout << "constructor called for balance " << balance << endl;   }

void account::getdetails(){
    cout<< " having rs.: " << balance  << endl;
}

int main(){
    account savings_acc1(1500);
    account savings_acc3 = savings_acc1+300;
    savings_acc3.getdetails();
    return 0;
}
```

Result:

```
constructor called for balance 1500
in operator bal: 1500
constructor called for balance 1800
 having rs.: 1800
Destructor called for object
Destructor called for object
```