#include<iostream>

using namespace std;

//class Test {

//public:

// int a;

// int b;

//};

//

//void main() {

// Test t1;

// t1.a = 100;

// t1.b = 200;

//

// //++t1;

// /\*if (t1 > t2) {

//

// }\*/

//

// //Test t3 = t1 + t2;

//

//

//

//}

//class Student {

// int age;

// double score;

//};

class INT {

int num;

public:

INT() :num(0) {}

INT(int num) {

SetNum(num);

}

int GetNum()const {

return num;

}

void SetNum(int num) {

this->num = num;

}

void Show()const {

cout << "Data : " << GetNum() << endl;

}

#pragma region Operator Overloading

//prefix

INT& operator++() {

this->num++;

return \*this;

}

//postfix

INT operator++(int) {

INT temp = \*this;

this->num++;

return temp;

}

//prefix

INT& operator--() {

this->num--;

return \*this;

}

//postfix

INT operator--(int) {

INT temp = \*this;

this->num--;

return temp;

}

INT& operator+=(const INT& other) {

this->num += other.GetNum();

return \*this;

}

bool operator>(const INT& other) {

if (this->num > other.num)

return true;

return false;

}

bool operator>=(const INT& other) {

if (this->num >= other.num)

return true;

return false;

}

bool operator==(const INT& other) {

if (this->num == other.num)

return true;

return false;

}

#pragma endregion

};

INT operator+(const INT& first, const INT& second) {

INT result(first.GetNum() + second.GetNum());

return result;

}

INT operator-(const INT& first, const INT& second) {

INT result(first.GetNum() - second.GetNum());

return result;

}

void main() {

//CAR id,name,year,engine

/\*INT a(100);

INT b(100);

INT temp = a + b;

temp.Show();

if (a == b) {

cout << "Okay" << endl;

}

else {

cout << "No" << endl;

}\*/

//a += b;

//a += 5;

a.Show();

//++a;

/\*a.Show();

cout << (a--).GetNum() << endl;

cout << (a).GetNum() << endl;\*/

//a.Show();

}