Experiment #1

1.

//Header1.h

#ifndef Header1\_h

#define Header1\_h

class Complex {

public:

Complex();

Complex(double[2]);

~Complex();

void Print();

void Add(Complex,Complex);

void Subtract(Complex, Complex);

private:

double cplx[2];

};

#endif // !Header1\_h

//////////////////////////////////////////////////

//Test.cpp

#include<iostream>

using namespace std;

#include"Header1.h"

Complex::Complex() {

cplx[0] = 0;

cplx[1] = 0;

}

Complex::Complex(double c[2]) {

cplx[0] = c[0];

cplx[1] = c[1];

}

Complex::~Complex() {

cout << '\"' << cplx[0] << " + " << cplx[1] << "i\" has been destructed.\n";

}//"x + xi" has been destructed.

void Complex::Print() {

cout << '(' << cplx[0] << " , " << cplx[1] << ')';

}//(x , x)

void Complex::Add(Complex c1, Complex c2) {

double result\_d[2] = { (c1.cplx[0] + c2.cplx[0]),(c1.cplx[1] + c2.cplx[1]) };

Complex result\_C(result\_d);

c1.Print();

cout << " + ";

c2.Print();

cout << " = ";

result\_C.Print();

cout << endl;

}//(x , x) + (x , x) = (x , x)

void Complex::Subtract(Complex c1, Complex c2) {

double result\_d[2] = { (c1.cplx[0] - c2.cplx[0]),(c1.cplx[1] - c2.cplx[1]) };

Complex result\_C(result\_d);

c1.Print();

cout << " - ";

c2.Print();

cout << " = ";

result\_C.Print();

cout << endl;

}//(x , x) - (x , x) = (x , x)

int main() {

double cplx\_1[2] = { 5,3 }, cplx\_2[2] = { -7,-9 };

Complex C1(cplx\_1);

Complex C2(cplx\_2);

Complex C3;

Complex result\_Add;

Complex result\_Subtract;

cout << "Print test:\nC1: ";

C1.Print();//Should be (5 , 3)

cout << "\nC2: ";

C2.Print();//Should be (-7 , -9)

cout << "\nC3(default): ";

C3.Print();//Should be (0 , 0)

cout << "\n\n\nAdd test:\n";

result\_Add.Add(C1, C2);//Should be (5 , 3) + (-7 , -9) = (-2 , -6)

cout << endl;

result\_Add.Add(C3, C1);//Should be (0 , 0) + (5 , 3) = (5 , 3)

cout << "\n\n\nSubstract test:\n";

result\_Subtract.Subtract(C1, C2);//Should be (5 , 3) - (-7 , -9) = (12 , 12)

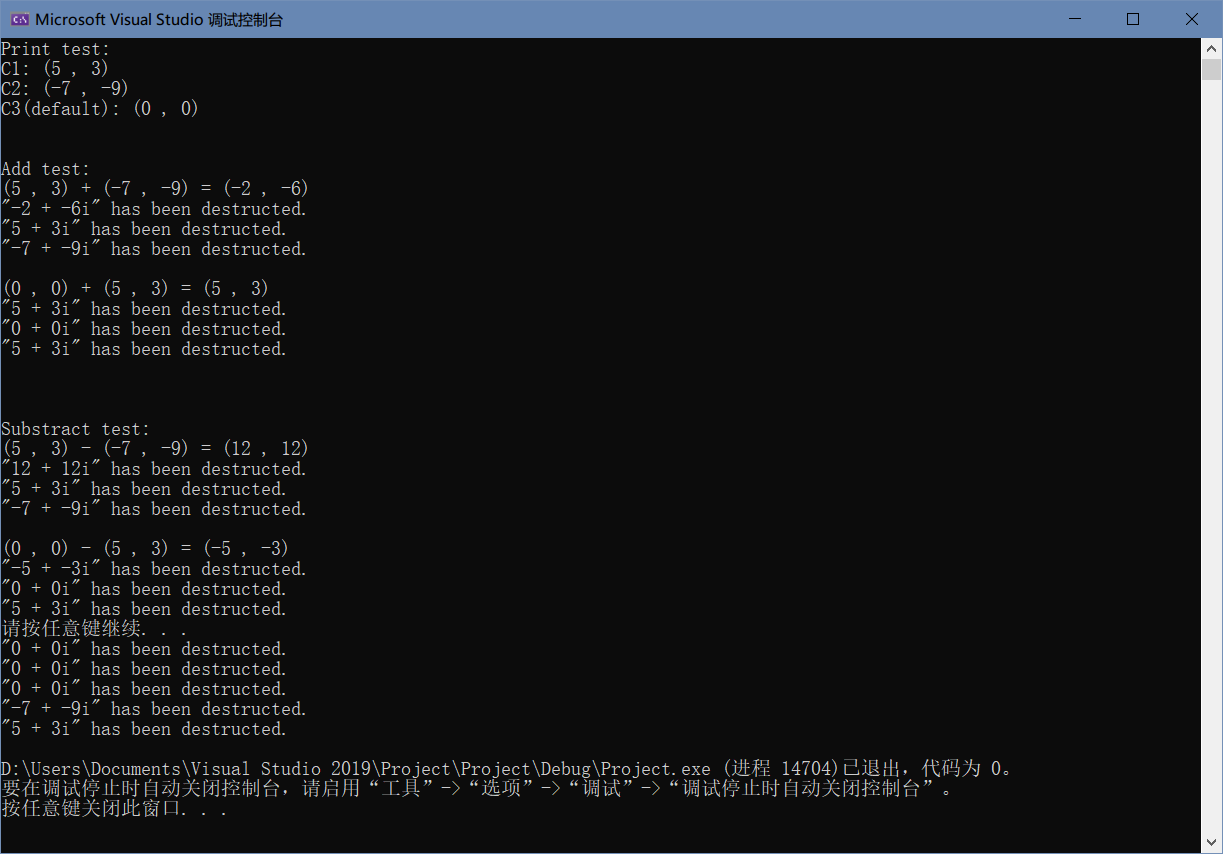
cout << endl;

result\_Subtract.Subtract(C3, C1);//Should be (0 , 0) - (5 , 3) = (-5 , -3)

system("pause");

return 0;

}



2.

//Header1.h

#ifndef Header\_h

#define Header\_h

class Time {

friend void tick(Time&);

public:

explicit Time(int = 0, int = 0, int = 0);

void setTime(int, int, int);

void setHour(int);

void setMinute(int);

void setSecond(int);

unsigned int getHour() const;

unsigned int getMinute() const;

unsigned int getSecond() const;

void printUniversal() const;

void printStandard() const;

private:

unsigned int hour, minute, second;

};

#endif // !Header\_h

//////////////////////////////////////////////////

//Test.cpp

#include<iostream>

#include<iomanip>

#include<stdexcept>

#include<Windows.h>

using namespace std;

#include"Header1.h"

void tick(Time& t) {

unsigned int s = t.getSecond();

unsigned int m = t.getMinute();

unsigned int h = t.getHour();

while (1) {

t.printStandard();

if (++s == 60) {

s -= 60;

if (++m == 60) {

m -= 60;

if (++h == 24)

h -= 24;

}

}

t.setTime(h, m, s);

Sleep(1000);

}

} // +1s and print in universal time

Time::Time(int hour, int minute, int second) {

setTime(hour, minute, second);

}

void Time::setTime(int h, int m, int s) {

setHour(h);

setMinute(m);

setSecond(s);

}

void Time::setHour(int h) {

if (h >= 0 && h < 24)

hour = h;

else

throw invalid\_argument("hour must be 0-23");

}

void Time::setMinute(int m) {

if (m >= 0 && m < 60)

minute = m;

else

throw invalid\_argument("minute must be 0-59");

}

void Time::setSecond(int s) {

if (s >= 0 && s < 60)

second = s;

else

throw invalid\_argument("second must be 0-59");

}

unsigned int Time::getHour() const {

return hour;

}

unsigned int Time::getMinute() const {

return minute;

}

unsigned int Time::getSecond() const {

return second;

}

void Time::printUniversal() const {

cout << setfill('0') << setw(2) << getHour() << ':'

<< setw(2) << getMinute() << ':' << setw(2) << getSecond() << endl;

}

void Time::printStandard() const {

cout << ((getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12)

<< ':' << setfill('0') << setw(2) << getMinute()

<< ':' << setw(2) << getSecond() << (hour < 12 ? " AM" : " PM") << endl;

}

int main() {

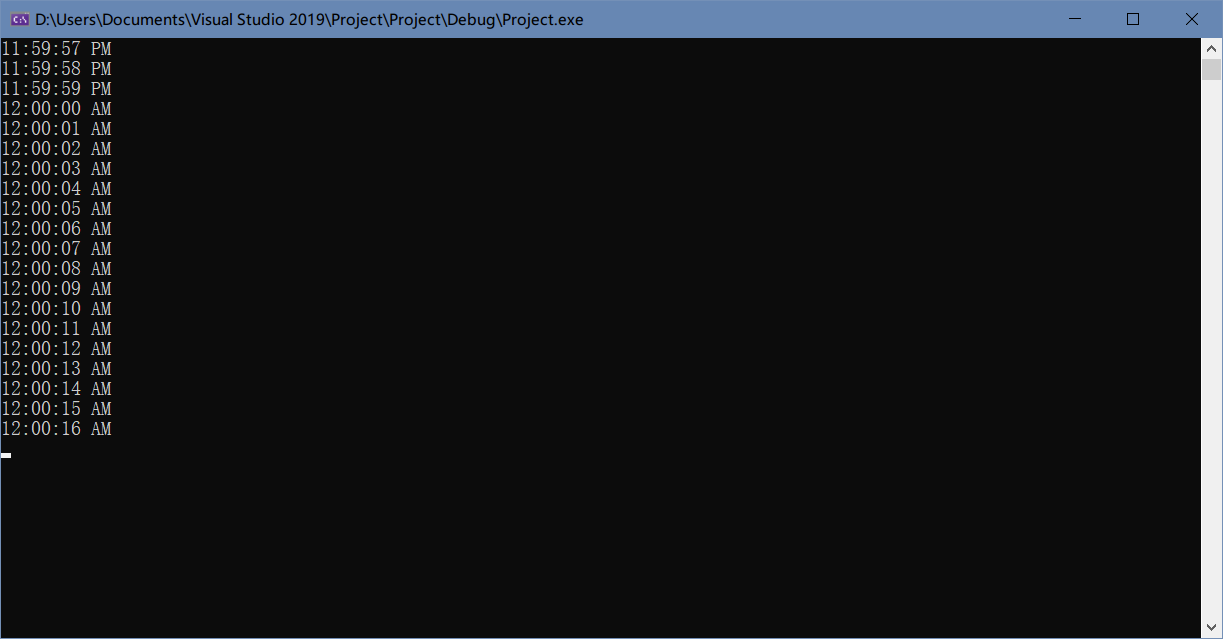
Time test(23, 59, 57);

tick(test);

system("pause");

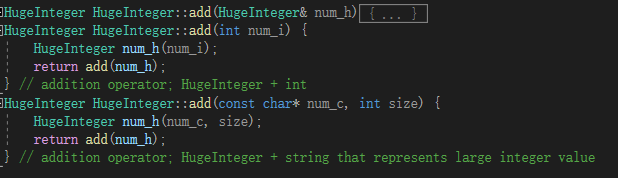
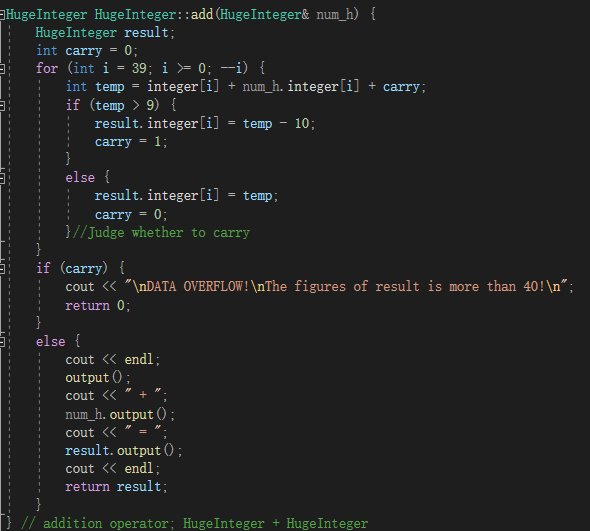
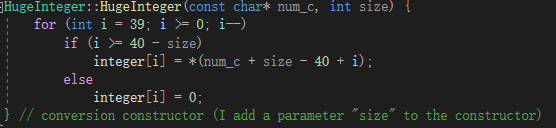
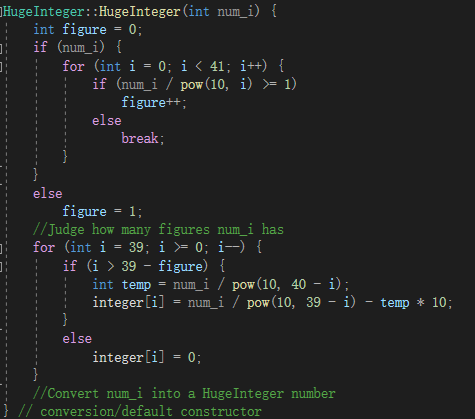
return 0;

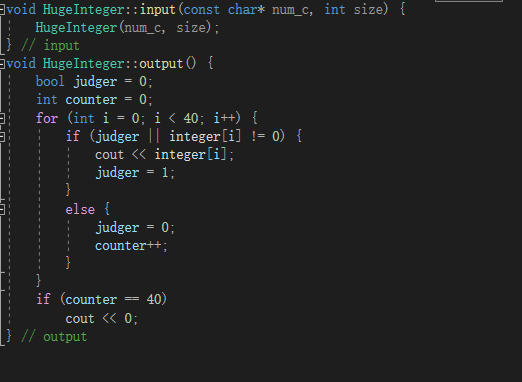
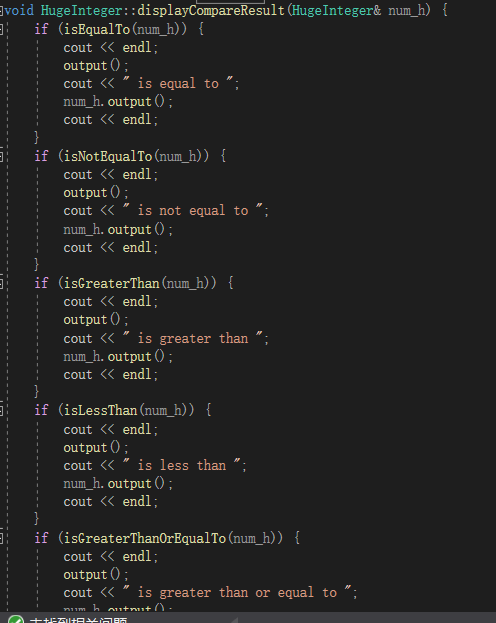
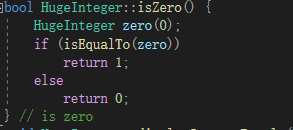
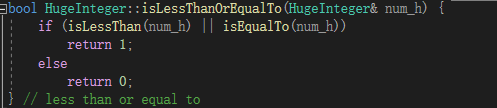
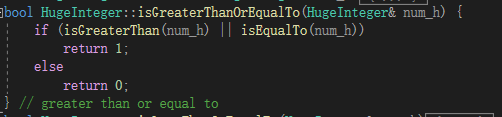
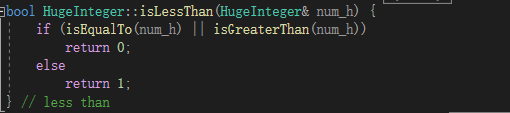
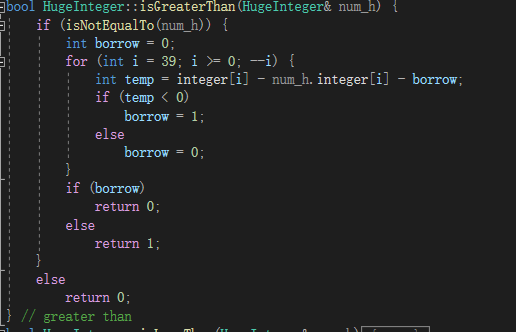
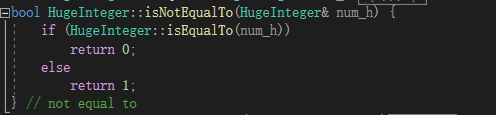
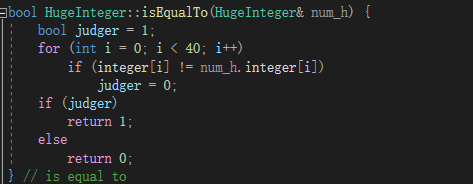
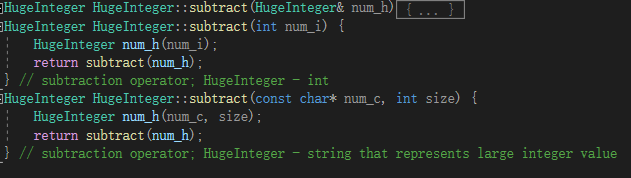
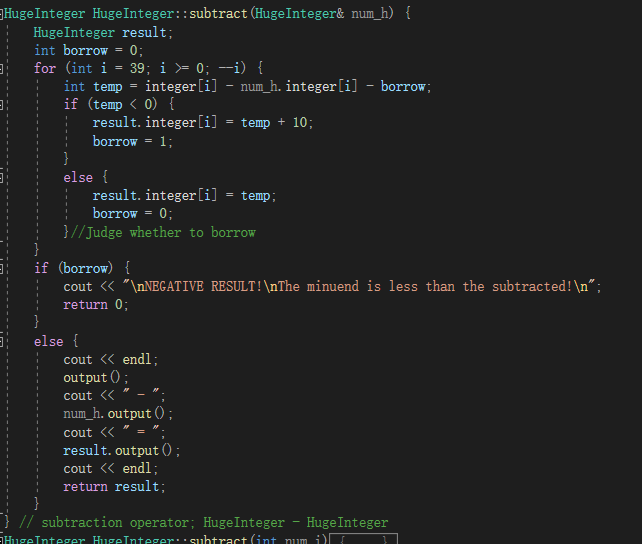
}



3.

//word读起来会很费劲，我就把重要的代码部分截了图按顺序放这了



//Header1.h

#ifndef HUGEINTEGER\_H

#define HUGEINTEGER\_H

class HugeInteger

{

public:

HugeInteger(int = 0);

HugeInteger(const char\*, int);

HugeInteger add(HugeInteger&);

HugeInteger add(int);

HugeInteger add(const char\*, int);

HugeInteger subtract(HugeInteger&);

HugeInteger subtract(int);

HugeInteger subtract(const char\*,int);

bool isEqualTo(HugeInteger&);

bool isNotEqualTo(HugeInteger&);

bool isGreaterThan(HugeInteger&);

bool isLessThan(HugeInteger&);

bool isGreaterThanOrEqualTo(HugeInteger&);

bool isLessThanOrEqualTo(HugeInteger&);

bool isZero();

void displayCompareResult(HugeInteger&);

void input(const char\*, int);

void output();

private:

int integer[40]; // 40 element array

}; // end class HugeInteger

#endif // !HUGEINTEGER\_H

//////////////////////////////////////////////////

//Test.cpp

#include<iostream>

#include<cmath>

#include<string>

using namespace std;

#include"Header1.h"

HugeInteger::HugeInteger(int num\_i) {

int figure = 0;

if (num\_i) {

for (int i = 0; i < 41; i++) {

if (num\_i / pow(10, i) >= 1)

figure++;

else

break;

}

}

else

figure = 1;

//Judge how many figures num\_i has

for (int i = 39; i >= 0; i--) {

if (i > 39 - figure) {

int temp = num\_i / pow(10, 40 - i);

integer[i] = num\_i / pow(10, 39 - i) - temp \* 10;

}

else

integer[i] = 0;

}

//Convert num\_i into a HugeInteger number

} // conversion/default constructor

HugeInteger::HugeInteger(const char\* num\_c, int size) {

for (int i = 39; i >= 0; i--)

if (i >= 40 - size)

integer[i] = \*(num\_c + size - 40 + i);

else

integer[i] = 0;

} // conversion constructor (I add a parameter "size" to the constructor)

HugeInteger HugeInteger::add(HugeInteger& num\_h) {

HugeInteger result;

int carry = 0;

for (int i = 39; i >= 0; --i) {

int temp = integer[i] + num\_h.integer[i] + carry;

if (temp > 9) {

result.integer[i] = temp - 10;

carry = 1;

}

else {

result.integer[i] = temp;

carry = 0;

}//Judge whether to carry

}

if (carry) {

cout << "\nDATA OVERFLOW!\nThe figures of result is more than 40!\n";

return 0;

}

else {

cout << endl;

output();

cout << " + ";

num\_h.output();

cout << " = ";

result.output();

cout << endl;

return result;

}

} // addition operator; HugeInteger + HugeInteger

HugeInteger HugeInteger::add(int num\_i) {

HugeInteger num\_h(num\_i);

return add(num\_h);

} // addition operator; HugeInteger + int

HugeInteger HugeInteger::add(const char\* num\_c, int size) {

HugeInteger num\_h(num\_c, size);

return add(num\_h);

} // addition operator; HugeInteger + string that represents large integer value

HugeInteger HugeInteger::subtract(HugeInteger& num\_h) {

HugeInteger result;

int borrow = 0;

for (int i = 39; i >= 0; --i) {

int temp = integer[i] - num\_h.integer[i] - borrow;

if (temp < 0) {

result.integer[i] = temp + 10;

borrow = 1;

}

else {

result.integer[i] = temp;

borrow = 0;

}//Judge whether to borrow

}

if (borrow) {

cout << "\nNEGATIVE RESULT!\nThe minuend is less than the subtracted!\n";

return 0;

}

else {

cout << endl;

output();

cout << " - ";

num\_h.output();

cout << " = ";

result.output();

cout << endl;

return result;

}

} // subtraction operator; HugeInteger - HugeInteger

HugeInteger HugeInteger::subtract(int num\_i) {

HugeInteger num\_h(num\_i);

return subtract(num\_h);

} // subtraction operator; HugeInteger - int

HugeInteger HugeInteger::subtract(const char\* num\_c, int size) {

HugeInteger num\_h(num\_c, size);

return subtract(num\_h);

} // subtraction operator; HugeInteger - string that represents large integer value

bool HugeInteger::isEqualTo(HugeInteger& num\_h) {

bool judger = 1;

for (int i = 0; i < 40; i++)

if (integer[i] != num\_h.integer[i])

judger = 0;

if (judger)

return 1;

else

return 0;

} // is equal to

bool HugeInteger::isNotEqualTo(HugeInteger& num\_h) {

if (HugeInteger::isEqualTo(num\_h))

return 0;

else

return 1;

} // not equal to

bool HugeInteger::isGreaterThan(HugeInteger& num\_h) {

if (isNotEqualTo(num\_h)) {

int borrow = 0;

for (int i = 39; i >= 0; --i) {

int temp = integer[i] - num\_h.integer[i] - borrow;

if (temp < 0)

borrow = 1;

else

borrow = 0;

}

if (borrow)

return 0;

else

return 1;

}

else

return 0;

} // greater than

bool HugeInteger::isLessThan(HugeInteger& num\_h) {

if (isEqualTo(num\_h) || isGreaterThan(num\_h))

return 0;

else

return 1;

} // less than

bool HugeInteger::isGreaterThanOrEqualTo(HugeInteger& num\_h) {

if (isGreaterThan(num\_h) || isEqualTo(num\_h))

return 1;

else

return 0;

} // greater than or equal to

bool HugeInteger::isLessThanOrEqualTo(HugeInteger& num\_h) {

if (isLessThan(num\_h) || isEqualTo(num\_h))

return 1;

else

return 0;

} // less than or equal to

bool HugeInteger::isZero() {

HugeInteger zero(0);

if (isEqualTo(zero))

return 1;

else

return 0;

} // is zero

void HugeInteger::displayCompareResult(HugeInteger& num\_h) {

if (isEqualTo(num\_h)) {

cout << endl;

output();

cout << " is equal to ";

num\_h.output();

cout << endl;

}

if (isNotEqualTo(num\_h)) {

cout << endl;

output();

cout << " is not equal to ";

num\_h.output();

cout << endl;

}

if (isGreaterThan(num\_h)) {

cout << endl;

output();

cout << " is greater than ";

num\_h.output();

cout << endl;

}

if (isLessThan(num\_h)) {

cout << endl;

output();

cout << " is less than ";

num\_h.output();

cout << endl;

}

if (isGreaterThanOrEqualTo(num\_h)) {

cout << endl;

output();

cout << " is greater than or equal to ";

num\_h.output();

cout << endl;

}

if (isLessThanOrEqualTo(num\_h)) {

cout << endl;

output();

cout << " is less than or equal to ";

num\_h.output();

cout << endl;

}

if (isZero()) {

cout << endl;

output();

cout << " is zero.\n";

}

} // I add a new function here to display all compare result

void HugeInteger::input(const char\* num\_c, int size) {

HugeInteger(num\_c, size);

} // input

void HugeInteger::output() {

bool judger = 0;

int counter = 0;

for (int i = 0; i < 40; i++) {

if (judger || integer[i] != 0) {

cout << integer[i];

judger = 1;

}

else {

judger = 0;

counter++;

}

}

if (counter == 40)

cout << 0;

} // output

int main() {

cout << "Caution:\na number of int type should be no more than 2147483647.\n\n";

int i\_1 = 987654321, i\_2 = 123456789;

char c\_1[] = { 0,9,8,7,6,5,4,3,2,1,2,3,4,5,6,7,8,9,0 }, c\_2[] = { 0,1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1,0 };

HugeInteger test\_output\_i\_1(i\_1);

HugeInteger test\_output\_i\_2(i\_2);

HugeInteger test\_output\_c\_1(c\_1, sizeof(c\_1));

HugeInteger test\_output\_c\_2(c\_2, sizeof(c\_2));

HugeInteger test\_output\_default;

char max[40];

for (int i = 0; i < 40; i++)

max[i] = 9;

HugeInteger test\_add\_h\_1(7654321);

HugeInteger test\_add\_h\_2(7891234);

HugeInteger test\_add\_h\_3(max, sizeof(max));

HugeInteger test\_add\_i\_1(7654321);

HugeInteger test\_add\_i\_2(max, sizeof(max));

HugeInteger test\_add\_c\_1;

HugeInteger test\_add\_c\_2(max, sizeof(max));

HugeInteger test\_subtract\_h\_1(7891234);

HugeInteger test\_subtract\_h\_2(1234567);

HugeInteger test\_subtract\_h\_3(9);

HugeInteger test\_subtract\_i\_1(7891234);

HugeInteger test\_subtract\_i\_2(9);

HugeInteger test\_subtract\_c\_1(max, sizeof(max));

HugeInteger test\_subtract\_c\_2(9);

HugeInteger test\_compare\_1(12345);

HugeInteger test\_compare\_2(12345);

HugeInteger test\_compare\_3(54321);

HugeInteger test\_compare\_4(0);

cout << "Output Test:";

cout << "\n1. i\_1: ";

test\_output\_i\_1.output();

cout << "\n2. i\_2: ";

test\_output\_i\_2.output();

cout << "\n3. c\_1: ";

test\_output\_c\_1.output();

cout << "\n4. c\_2: ";

test\_output\_c\_2.output();

cout << "\n5. Default: ";

test\_output\_default.output();

cout << endl << endl;

cout << "Adding Test:";

cout << "\n1. HugeInteger + HugeInteger:";

test\_add\_h\_1.add(test\_add\_h\_2);

test\_add\_h\_3.add(test\_add\_h\_2);

cout << "\n2. HugeInteger + int:";

test\_add\_i\_1.add(7891234);

test\_add\_i\_2.add(7891234);

cout << "\n3. HugeInteger + string:";

test\_add\_c\_1.add(test\_add\_c\_2);

cout << endl << endl;

cout << "Subtracting Test:";

cout << "\n1. HugeInteger - HugeInteger:";

test\_subtract\_h\_1.subtract(test\_subtract\_h\_2);

test\_subtract\_h\_3.subtract(test\_subtract\_h\_2);

cout << "\n2. HugeInteger - int:";

test\_subtract\_i\_1.subtract(1234567);

test\_subtract\_i\_2.subtract(1234567);

cout << "\n3. HugeInteger - string:";

test\_subtract\_c\_1.subtract(test\_subtract\_c\_2);

cout << endl << endl;

cout << "Comparing Test:";

cout << "\n1.\n";

test\_compare\_1.displayCompareResult(test\_compare\_2);

cout << "\n2.\n";

test\_compare\_3.displayCompareResult(test\_compare\_4);

cout << "\n3.\n";

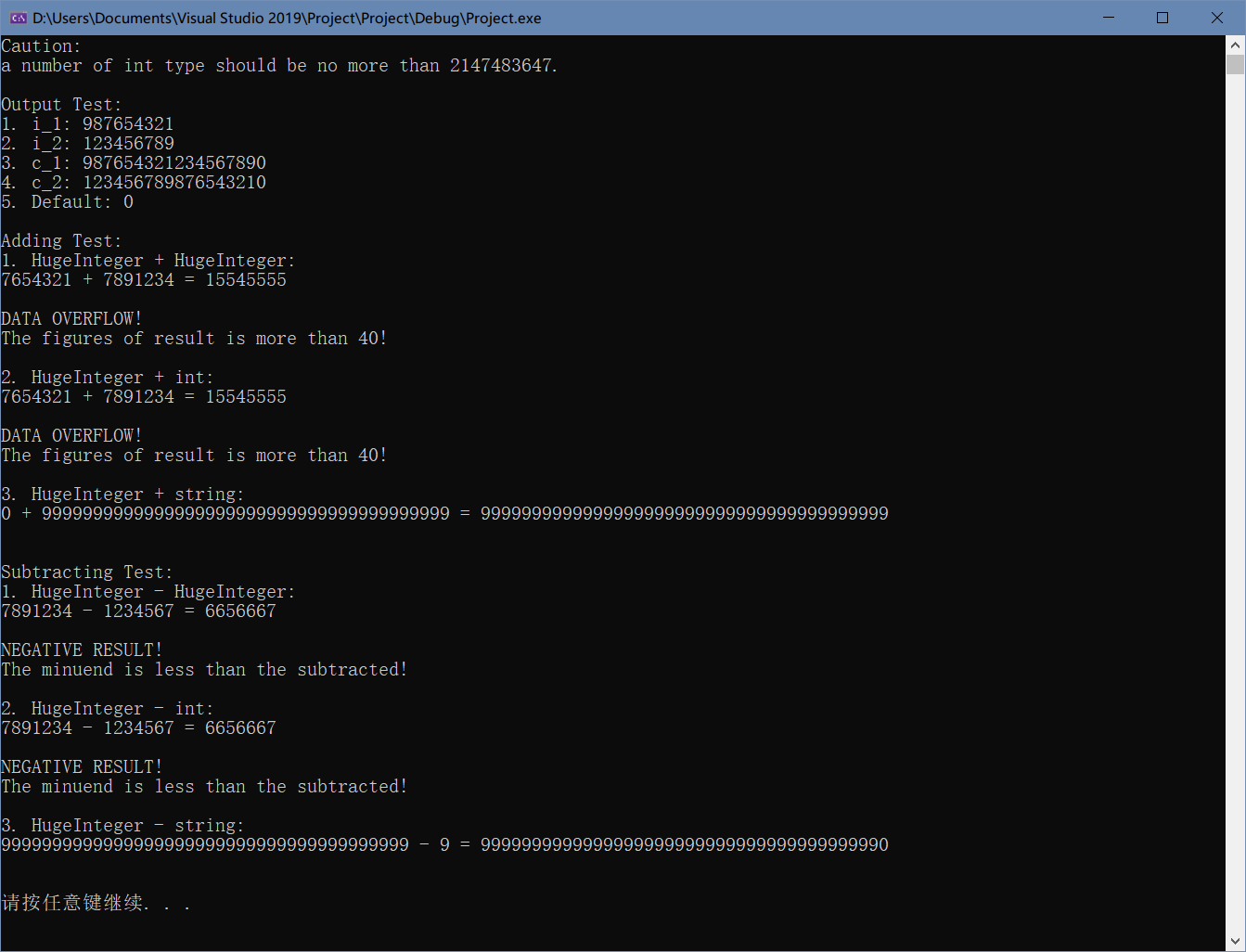
test\_compare\_4.displayCompareResult(test\_compare\_1);

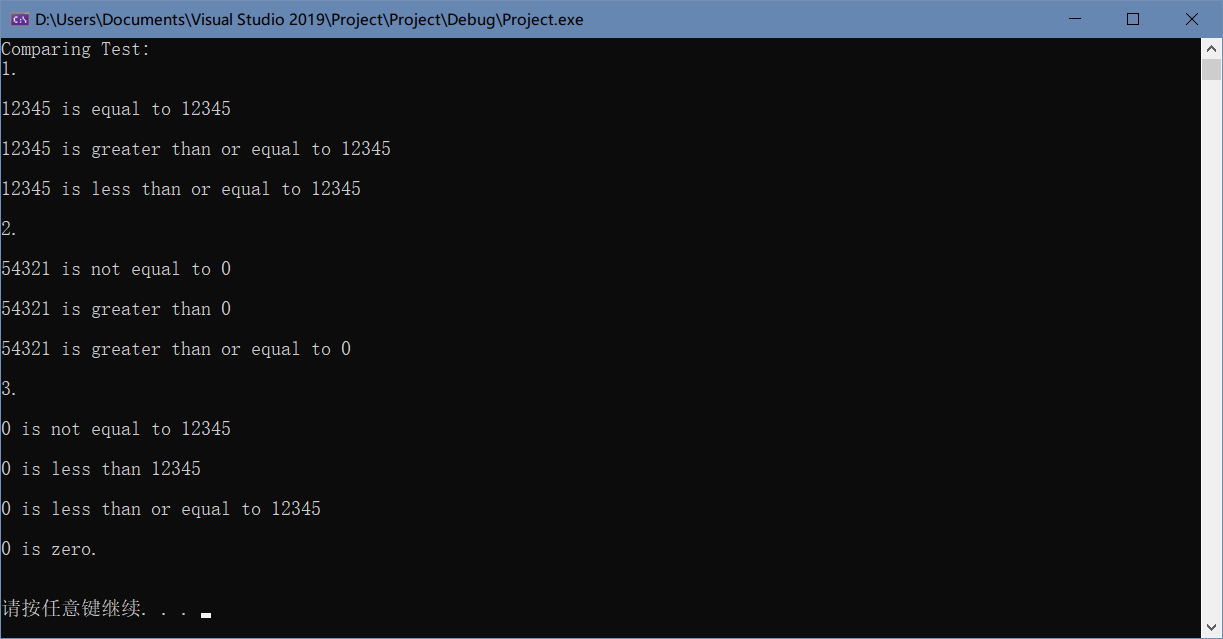
cout << endl << endl;

system("pause");

return 0;

}





4.

//Header1.h

#ifndef Header\_h

#define Header\_h

class SimpleCaculator {

public:

SimpleCaculator(double = 0, double = 0);

~SimpleCaculator();

double Add(double, double);

double Subtract(double, double);

double Multiple(double, double);

double Divide(double, double);

void input(double = 0, double = 0);

private:

double a, b;

};

#endif // !Header\_h

//////////////////////////////////////////////////

//Test.cpp

#include<iostream>

using namespace std;

#include"Header1.h"

SimpleCaculator::SimpleCaculator(double num1, double num2) {

input(num1, num2);

cout << "The value of a is: " << a

<< "\nThe value of b is: " << b << endl << endl;

}

SimpleCaculator::~SimpleCaculator() {

cout << "\"SimpleCaculator\" has been destructed.\n";

}

double SimpleCaculator::Add(double num1, double num2) {

input(num1, num2);

cout << "Adding a and b yields " << (a + b) << endl;

return (a + b);

}

double SimpleCaculator::Subtract(double num1, double num2) {

input(num1, num2);

cout << "Subtracting a and b yields " << (a - b) << endl;

return (a - b);

}

double SimpleCaculator::Multiple(double num1, double num2) {

input(num1, num2);

cout << "Multiplying a and b yields " << (a \* b) << endl;

return (a \* b);

}

double SimpleCaculator::Divide(double num1, double num2) {

input(num1, num2);

cout << "Dividing a and b yields " << (a / b) << endl;

return (a / b);

}

void SimpleCaculator::input(double num1, double num2) {

a = num1;

b = num2;

}

int main() {

double a = 10, b = 20;

SimpleCaculator sc(a, b);

double answer\_add = sc.Add(a, b);

double answer\_subtract = sc.Subtract(a, b);

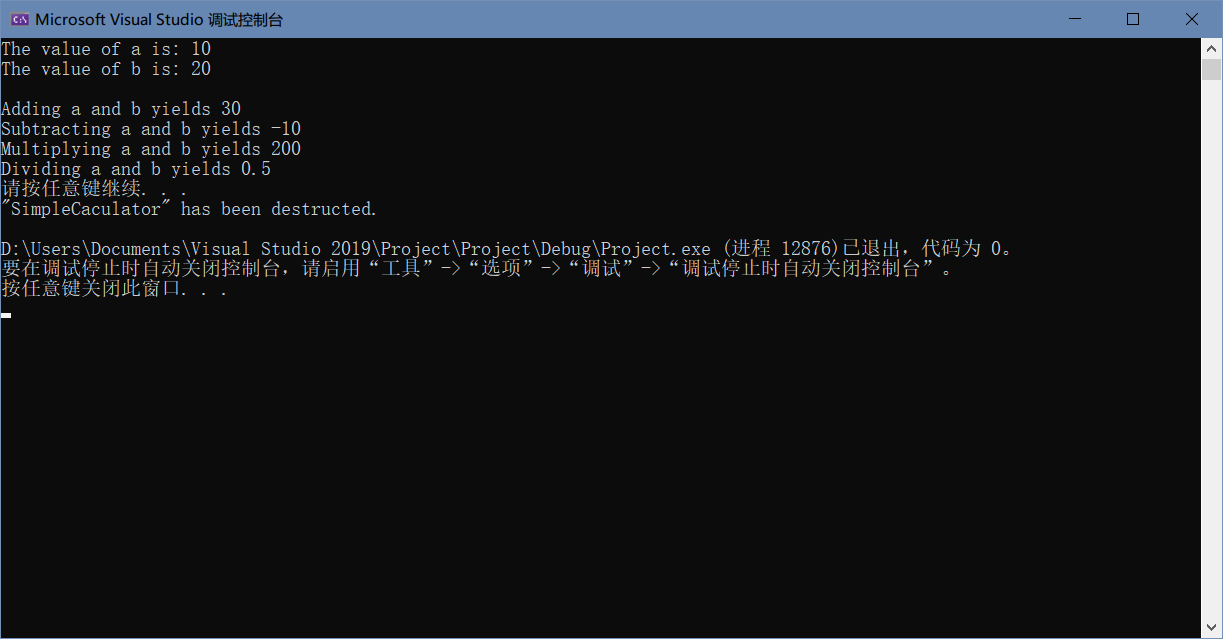
double answer\_multiple = sc.Multiple(a, b);

double answer\_divide = sc.Divide(a, b);

system("pause");

return 0;

}



5.

//Header1.h

#ifndef Header\_h

#define Header\_h

class IntegerSet {

public:

IntegerSet();

IntegerSet(int[], int);

~IntegerSet();

void initializeSet();

void inputSet(int[], int);

void printSet();

IntegerSet unionOfSets(IntegerSet, IntegerSet);

IntegerSet intersectionOfSets(IntegerSet, IntegerSet);

void insertElement(int);

void insertElement(int[], int);

void deleteElement(int);

void deleteElement(int[], int);

bool isEqualTo(IntegerSet);

private:

bool set[101];

};

#endif // !Header\_h

//////////////////////////////////////////////////

//Test.cpp

#include<iostream>

#include<iomanip>

using namespace std;

#include"Header1.h"

IntegerSet::IntegerSet() {

initializeSet();

}//defalut constructor

IntegerSet::IntegerSet(int input[], int size) {

inputSet(input, size);

}//constructor with an array as parameter

IntegerSet::~IntegerSet() {}//default destructor

void IntegerSet::initializeSet() {

for (int i = 0; i <= 100; i++)

set[i] = 0;

}//set all element to 0

void IntegerSet::inputSet(int input[], int size) {

initializeSet();

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

if (input[i] >= 0 && input[i] <= 100)

set[input[i]] = 1;

else

cout << "Invalid input: " << input[i] << endl;

}

}//input

void IntegerSet::printSet() {

IntegerSet zero;

if (isEqualTo(zero))

cout << "{ 0 }\n\n";

else {

cout << '{';

for (int i = 0; i <= 100; i++) {

if (set[i])

cout << setw(4) << i;

}

cout << " }\n\n";

}

}//output

IntegerSet IntegerSet::unionOfSets(IntegerSet setA, IntegerSet setB) {

IntegerSet setU;

for (int i = 0; i <= 100; i++) {

if (setA.set[i] || setB.set[i])

setU.set[i] = 1;

}

return setU;

}

IntegerSet IntegerSet::intersectionOfSets(IntegerSet setA, IntegerSet setB) {

IntegerSet setI;

for (int i = 0; i <= 100; i++) {

if (setA.set[i] && setB.set[i])

setI.set[i] = 1;

}

return setI;

}

void IntegerSet::insertElement(int insert) {

if (insert >= 0 && insert <= 100)

set[insert] = 1;

else

cout << "Invalid input: " << insert << endl;

}//insert only a number

void IntegerSet::insertElement(int insert[], int size) {

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

if (insert[i] >= 0 && insert[i] <= 100)

set[insert[i]] = 1;

else

cout << "Invalid input: " << insert[i] << endl;

}

}//insert an array

void IntegerSet::deleteElement(int insert) {

if (insert >= 0 && insert <= 100)

set[insert] = 0;

else

cout << "Invalid input: " << insert << endl;

}//delete a number

void IntegerSet::deleteElement(int insert[], int size) {

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

if (insert[i] >= 0 && insert[i] <= 100)

set[insert[i]] = 0;

else

cout << "Invalid input: " << insert[i] << endl;

}

}//delete an array

bool IntegerSet::isEqualTo(IntegerSet input) {

bool result = 1;

for (int i = 0; i <= 100; i++) {

if (set[i] != input.set[i]) {

result = 0;

break;

}

else

continue;

}

return result;

}

int main() {

int A[6] = { 45,76,34,6,-1,101 };

int B[6] = { 34,8,93,45,-1,101 };

IntegerSet setA(A, sizeof(A) / 4);

IntegerSet setB;

IntegerSet setC;

IntegerSet setD;

cout << "Set A is:\n";

setA.printSet();

cout << "Set B is:\n";

setB.printSet();

cout << "Input { 34,8,93,45,-1,101 } into set B.\n";

setB.inputSet(B, sizeof(B) / 4);

cout << "Now set B is:\n";

setB.printSet();

cout << "Union of A and B is:\n";

setC = setC.unionOfSets(setA, setB);

setC.printSet();

cout << "Intersection of A and B is:\n";

setD = setD.intersectionOfSets(setA, setB);

setD.printSet();

if (setA.isEqualTo(setB))

cout << "Set A is equal to B.\n\n";

else

cout << "Set A is not equal to B.\n\n";

cout << "Insert { -5,200,77 } into set A.\n";

int ist[3] = { -5,200,77 };

setA.insertElement(ist, 3);

cout << "Now set A is:\n";

setA.printSet();

cout << "Delete { -123,456,77 } from set A.\n";

int dlt[3] = { -123,456,77 };

setA.deleteElement(dlt, 3);

cout << "Now set A is:\n";

setA.printSet();

system("pause");

return 0;

}

