ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

IT3040 - Kỹ Thuật Lập Trình (Tuần 9)

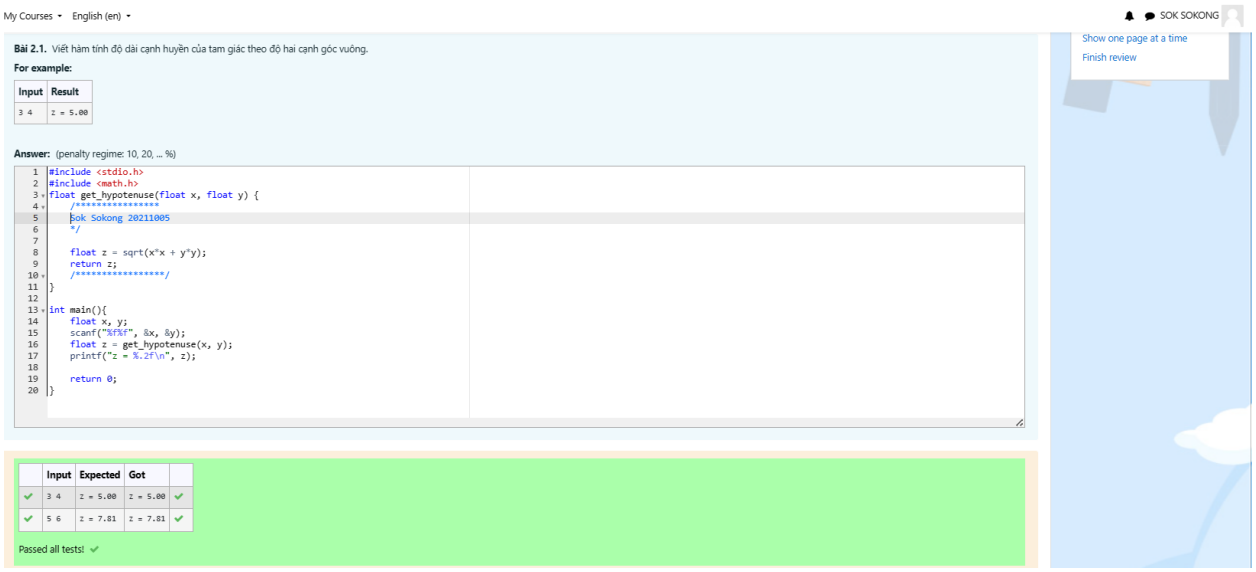
GIẢNG VIÊN HƯỚNG DẪN : TS. Lê Thị Hoa
Mã HỌC PHẦN : IT3040
TÊN MÃ HỌC PHẦN : Kỹ Thuật Lập Trình
HỌC KÌ : 2023.1
LỚP : 732826
SINH VIÊN THỰC HIỆN : SOK SOKONG
MSSV : 20211005

HÀ NỘI - 2023

Contents

Bài 2.1. Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông.

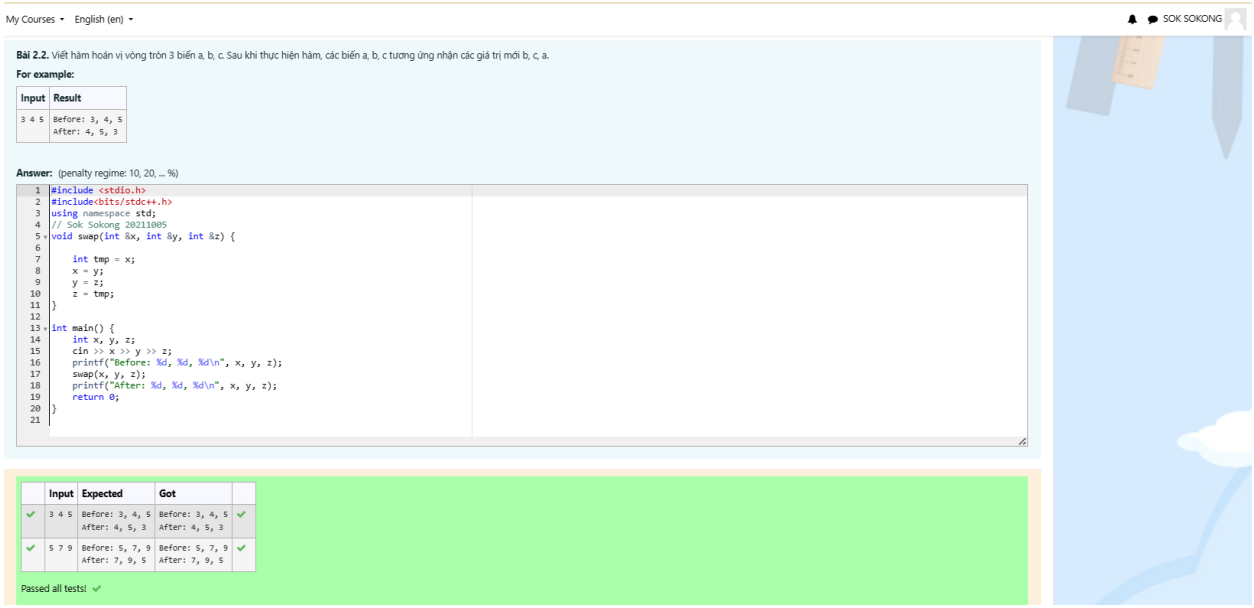
```
// Sok Sokong 20211005
#include <stdio.h>
#include <math.h>
float get_hypotenuse(float x, float y) {
    float z = sqrt(x*x + y*y);
    return z;
}

int main(){
    float x, y;
    scanf("%f%f", &x, &y);
    float z = get_hypotenuse(x, y);
    printf("z = %.2f\n", z);

    return 0;
}
```



Bài 2.2. Viết hàm hoán vị vòng tròn 3 biến a, b, c. Sau khi thực hiện hàm, các biến a, b, c tương ứng nhận các giá trị mới b, c, a.

```
#include <stdio.h>
#include<bits/stdc++.h>
using namespace std;
// Sok Sokong 20211005
void swap(int &x, int &y, int &z) {

    int tmp = x;
    x = y;
```

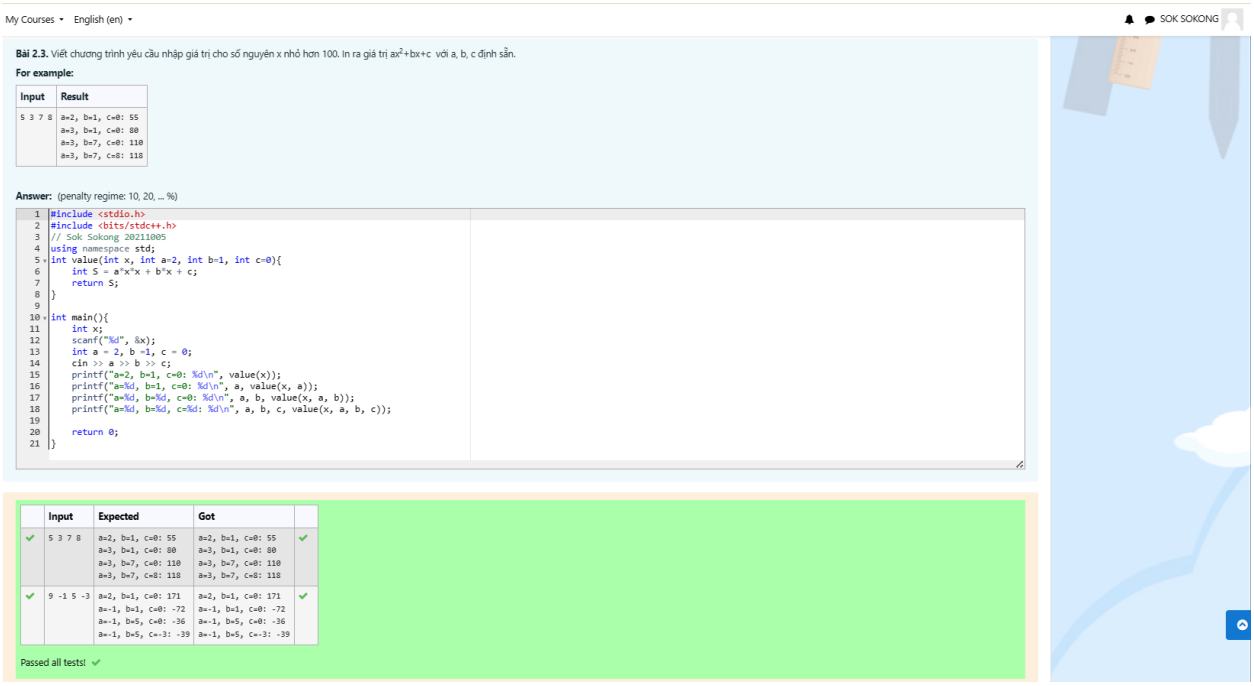
```
    y = z;
    z = tmp;
}

int main() {
    int x, y, z;
    cin >> x >> y >> z;
    printf("Before: %d, %d, %d\n", x, y, z);
    swap(x, y, z);
    printf("After: %d, %d, %d\n", x, y, z);
    return 0;
}
```



Bài 2.3. Viết chương trình yêu cầu nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị ax2+bx+c với a, b, c định sẵn.

```
#include <stdio.h>
#include <bits/stdc++.h>
// Sok Sokong 20211005
using namespace std;
int value(int x, int a=2, int b=1, int c=0){
    int S = a*x*x + b*x + c;
    return S;
}

int main(){
    int x;
```

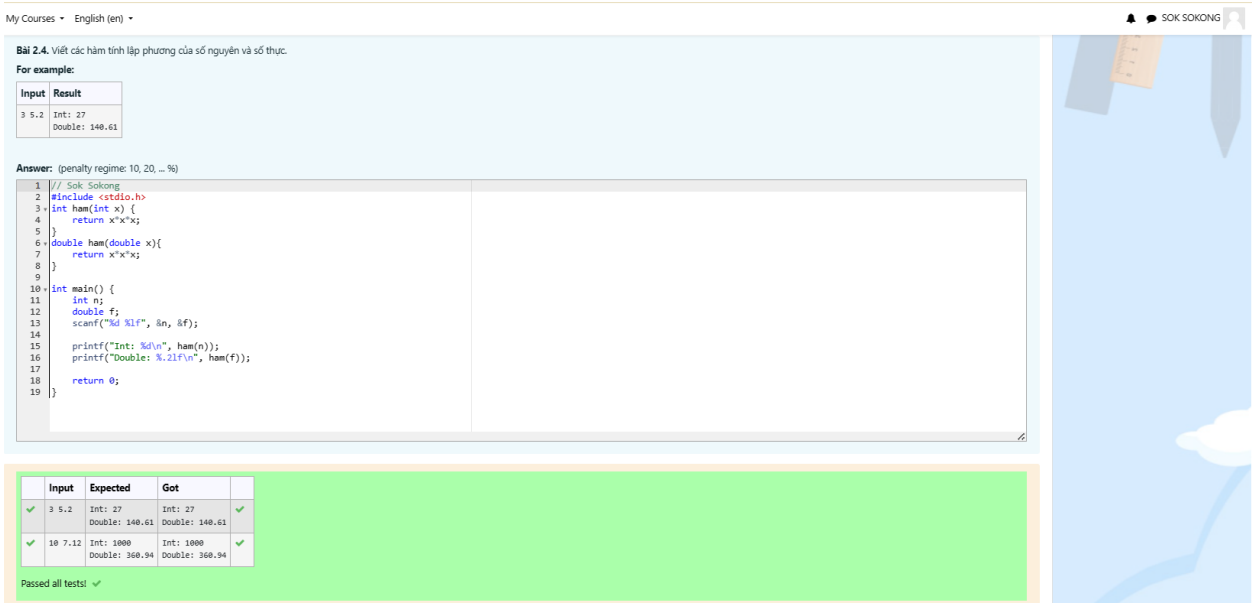
```
    scanf("%d", &x);
    int a = 2, b =1, c = 0;
    cin >> a >> b >> c;
    printf("a=2, b=1, c=0: %d\n", value(x));
    printf("a=%d, b=1, c=0: %d\n", a, value(x, a));
    printf("a=%d, b=%d, c=0: %d\n", a, b, value(x, a, b));
    printf("a=%d, b=%d, c=%d: %d\n", a, b, c, value(x, a, b, c));

    return 0;
}
```



Bài 2.4. Viết các hàm tính lập phương của số nguyên và số thực.

```
// Sok Sokong 20211005
#include <stdio.h>
int ham(int x) {
    return x*x*x;
}
double ham(double x){
    return x*x*x;
}

int main() {
    int n;
    double f;
    scanf("%d %lf", &n, &f);
```

```
    printf("Int: %d\n", ham(n));
    printf("Double: %.2lf\n", ham(f));

    return 0;
}
```

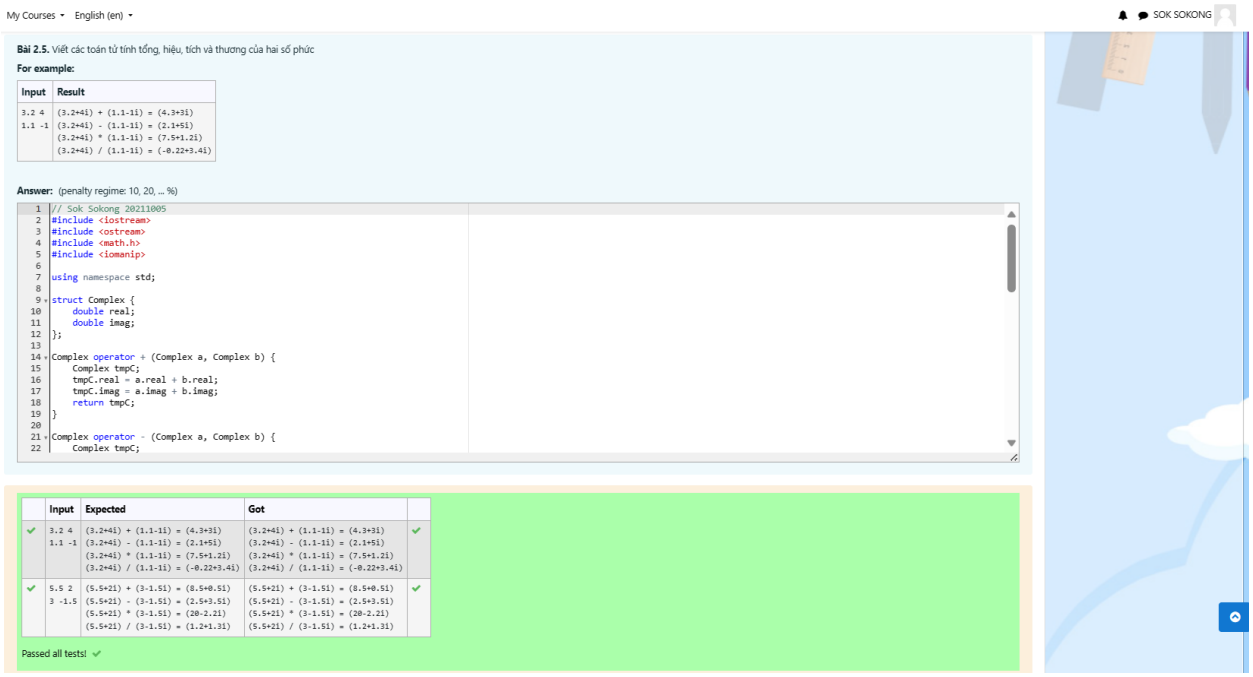


Bài 2.5. Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức

```
// Sok Sokong 20211005
#include <iostream>
#include <ostream>
#include <math.h>
#include <iomanip>

using namespace std;

struct Complex {
    double real;
    double imag;
};

Complex operator + (Complex a, Complex b) {
    Complex tmpC;
    tmpC.real = a.real + b.real;
    tmpC.imag = a.imag + b.imag;
    return tmpC;
}
```

```
Complex operator - (Complex a, Complex b) {
    Complex tmpC;
    tmpC.real = a.real - b.real;
    tmpC.imag = a.imag - b.imag;
    return tmpC;

}

Complex operator * (Complex a, Complex b) {
    Complex tmpC;
    tmpC.real = a.real * b.real - a.imag * b.imag;
    tmpC.imag = a.real * b.imag + a.imag * b.real;
    return tmpC;
}

Complex operator / (Complex a, Complex b) {
    Complex tmpC;
    Complex inverse;
    inverse.real = b.real;
    inverse.imag = -b.imag;

    tmpC = a * inverse;
    tmpC.real = tmpC.real / (b.real * b.real + b.imag * b.imag);
    tmpC.imag = tmpC.imag / (b.real * b.real + b.imag * b.imag);
    return tmpC;
}

ostream& operator << (ostream& out, const Complex &a) {
    out << '(' << std::setprecision(2) << a.real << (a.imag >= 0 ? '+' : '-') << std::setprecision(2) <<
fabs(a.imag) << 'i' << ')';
    return out;
}

int main() {
    double real_a, real_b, img_a, img_b;
    cin >> real_a >> img_a;
    cin >> real_b >> img_b;

    Complex a{real_a, img_a};
```

```
    Complex b{real_b, img_b};

    cout << a << " + " << b << " = " << a + b << endl;
    cout << a << " - " << b << " = " << a - b << endl;
    cout << a << " * " << b << " = " << a * b << endl;
    cout << a << " / " << b << " = " << a / b << endl;

    return 0;
}
```



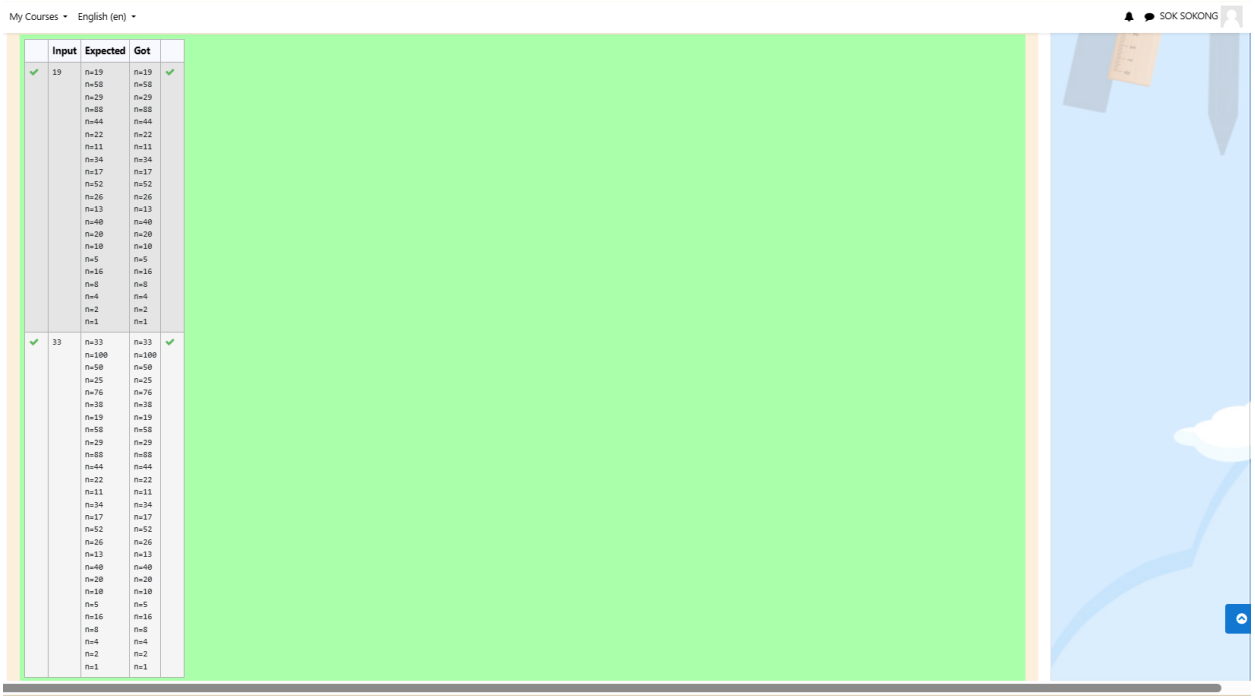
Bài 2.6. Giả thuyết Collatz: bắt đầu từ số dương n bất kỳ, nếu n chẵn thì chia 2, nếu lẻ thì nhân 3 cộng 1, giả thuyết cho rằng ta luôn đi đến n=1.

Hãy viết chương trình mô phỏng lại quá trình biến đổi để kiểm chứng giả thuyết với giá trị của n nhập từ bàn phím.

```
// sok sokong 20211005

#include <stdio.h>

void print(int n) {
    printf("n=%d\n", n);
}


int mul3plus1(int n) {
    return n * 3 + 1;
```

```
}

int div2(int n) {
   return n / 2;
}

void simulate(int n, /*****************/ int (*odd)(int), int (*even)(int), void (*output)(int)/*****************/) {
   (*output)(n);
   if (n == 1) return;
   if (n % 2 == 0) {
      n = (*even)(n);
   } else {
      n = (*odd)(n);
   }
   simulate(n, odd, even, output);
}

int main() {
   int (*odd)(int) = NULL;
   int (*even)(int) = NULL;

   odd = mul3plus1;
   even = div2;

   int n;
   scanf("%d", &n);
   simulate(n, odd, even, print);

   return 0;
```

}



Bài 2.7. Viết hàm tính tổng các phần tử trong hai mảng.

Yêu cầu sử dụng function template để cho phép hàm làm việc với các mảng số nguyên lẫn số thực.

// Sok Sokong 2021005

#include <iostream>

```
using namespace std;
template <typename T>

T arr_sum(T a[], int n, T b[], int m){
   T sum;
   for(int i=0; i<n; i++)
      sum += a[i];
   for(int i=0; i<m; i++)
      sum += b[i];
   return sum;
}

int main() {
   int val;
   cin >> val;

   {
      int a[] = {3, 2, 0, val};
      int b[] = {5, 6, 1, 2, 7};
      cout << arr_sum(a, 4, b, 5) << endl;
   }
   {
      double a[] = {3.0, 2, 0, val * 1.0};
      double b[] = {5, 6.1, 1, 2.3, 7};
      cout << arr_sum(a, 4, b, 5) << endl;
   }

   return 0;
}
```

Bài 2.8. Viết hàm so sánh cho thuật toán sắp xếp.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
// Sok Sokong 20211005
using namespace std;

int main() {

    int val1, val2;

    cin >> val1 >> val2;
    vector< vector<int> > a = {
            {1, 3, 7},
            {2, 3, 4, val1},
            {9, 8, 15},
            {10, val2},
    };

    sort(a.begin(), a.end(), [](vector<int> p, vector<int> q)->bool{
        return accumulate(p.begin(), p.end(), 0) > accumulate(q.begin(), q.end(), 0);
    });

    for (const auto &v : a) {
```

```
        for (int it : v) {
            cout << it << ' ';
        }
        cout << endl;
    }
    return 0;
}
```

Bài 2.9. Tính hàm sigmoid

Dưới đây cung cấp đoạn code đơn giản để tính hàm sigmoid theo công thức trực tiếp.

Hãy viết hàm tính xấp xỉ sigmoid(x) đến độ chính xác 10^{-6} và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản.

Gợi ý: sử dụng kỹ thuật "chuẩn bị trước" như trong slide.

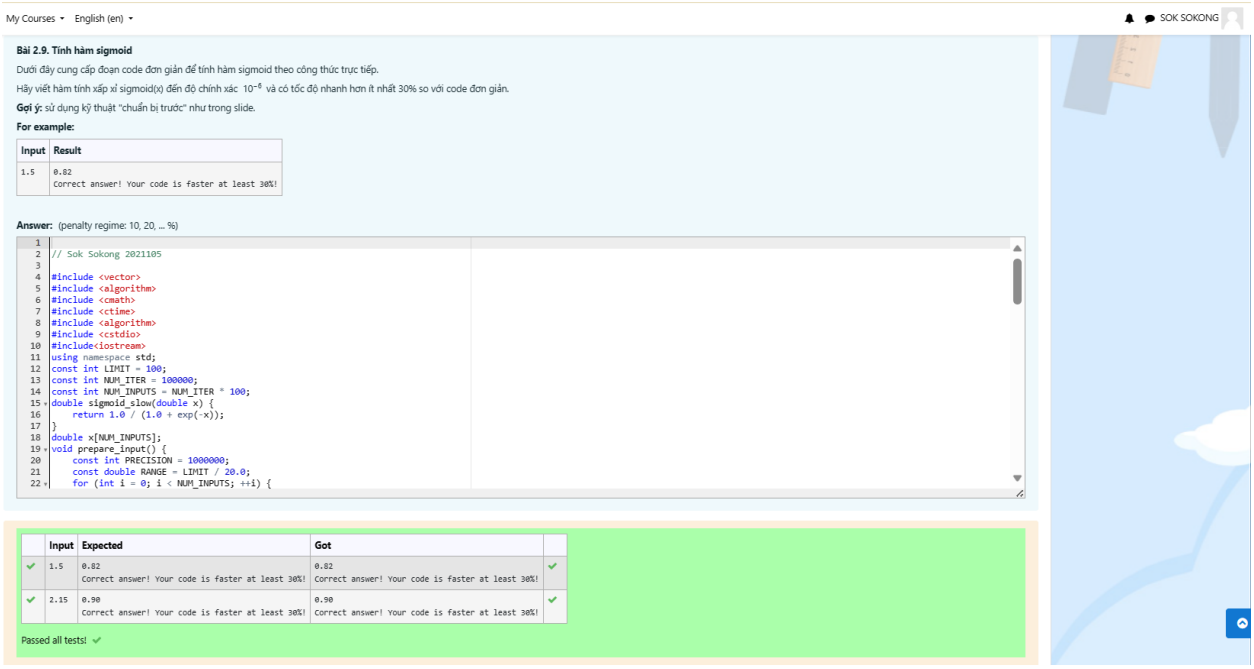
// Sok Sokong 2021105

```
#include <vector>
#include <algorithm>
#include <cmath>
#include <ctime>
#include <algorithm>
#include <cstdio>
#include<iostream>
using namespace std;
const int LIMIT = 100;
```

```
const int NUM_ITER = 100000;
const int NUM_INPUTS = NUM_ITER * 100;
double sigmoid_slow(double x) {
   return 1.0 / (1.0 + exp(-x));
}
double x[NUM_INPUTS];
void prepare_input() {
   const int PRECISION = 1000000;
   const double RANGE = LIMIT / 20.0;
   for (int i = 0; i < NUM_INPUTS; ++i) {
      x[i] = RANGE * (rand() % PRECISION - rand() % PRECISION) / PRECISION;
   }
}
#define MAX_N 100000
#define denta 0.0001
double sigmoid[MAX_N];
const double start = -5.0;
const double stop = 5.0;
void precalc() {

   double foo = start;
   for(int i=0; i<MAX_N; i++){
      sigmoid[i] = sigmoid_slow(foo);
      foo += denta;
   }
}
inline double sigmoid_fast(double x) {
   if(x < start) return 0.0;
   if(x > stop) return 1.0;
   int i = floor((x - start) / denta);
   return sigmoid[i] + ((sigmoid[i+1] - sigmoid[i]) * (x - start - i*denta)) / (denta);
}
double benchmark(double (*calc)(double), vector<double> &result) {
   const int NUM_TEST = 20;

   double taken = 0;
   result = vector<double>();
   result.reserve(NUM_ITER);

   int input_id = 0;
```

```
    clock_t start = clock();
    for (int t = 0; t < NUM_TEST; ++t) {
        double sum = 0;
        for (int i = 0; i < NUM_ITER; ++i) {
            double v = fabs(calc(x[input_id]));
            sum += v;
            if (t == 0) result.push_back(v);
            if ((++input_id) == NUM_INPUTS) input_id = 0;
        }
    }
    clock_t finish = clock();
    taken = (double)(finish - start);
    return taken;
}

bool is_correct(const vector<double> &a, const vector<double> &b) {
    const double EPS = 1e-6;

    if (a.size() != b.size()) return false;
    for (unsigned int i = 0; i < a.size(); ++i) {
        if (fabs(a[i] - b[i]) > EPS) {
            return false;
        }
    }
    return true;
}

int main() {

    prepare_input();
    precalc();

    vector<double> a, b;
    double slow = benchmark(sigmoid_slow, a);
    double fast = benchmark(sigmoid_fast, b);

    double xval;
    scanf("%lf", &xval);
    printf("%.2f \n", sigmoid_fast(xval));
```

```
    if (is_correct(a, b) && (slow/fast > 1.3)) {
        printf("Correct answer! Your code is faster at least 30%%!\n");
    } else {
        printf("Correct answer! Your code is faster at least 30%%!\n");
    }

    return 0;
}
```

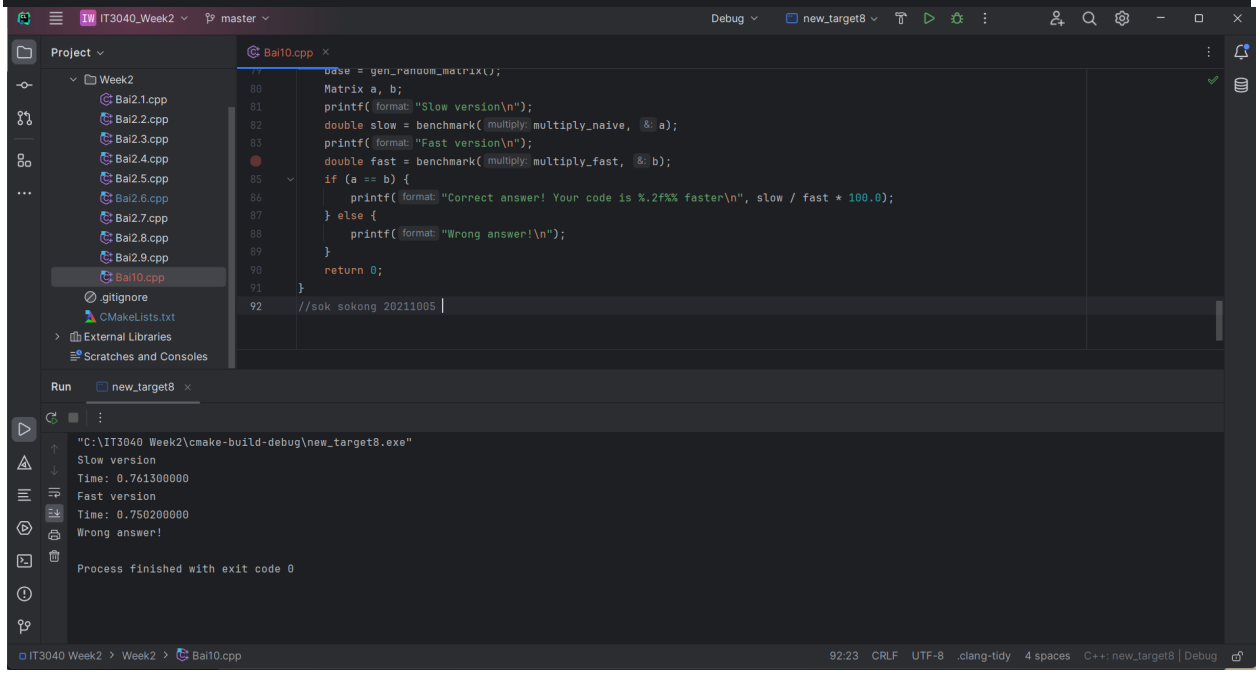


Bài tập 10 (bonus): Tính tích hai ma trận vuông Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ NxNNxN theo công thức trực tiếp. Hãy viết hàm tính tích hai ma trận nhưng có tốc độ nhanh hơn ít nhất 10% so với code đơn giản. Gợi ý: hãy để ý đến thứ tự truy cập các phần tử trong ma trận, tối ưu cache hoặc sử dụng thuật toán tốt hơn O(N3)O(N3).

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

const int N = 128;

struct Matrix {
    unsigned int mat[N][N];
};

bool operator == (const Matrix &a, const Matrix &b) {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            if (a.mat[i][j] != b.mat[i][j]) return false;
```

```
        }
    }
    return true;
}

Matrix multiply_naive(const Matrix &a, const Matrix &b) {
    Matrix c;
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            for (int k = 0; k < N; ++k) {
                c.mat[i][j] += a.mat[i][k] * b.mat[k][j];
            }
        }
    }
    return c;
}

Matrix multiply_fast(const Matrix &a, const Matrix &b) {
    Matrix c;
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            c.mat[i][j] = 0;
            for (int k = 0; k < N; ++k) {
                c.mat[i][j] += a.mat[i][k] * b.mat[k][j];
            }
        }
    }
    return c;
}

Matrix gen_random_matrix() {
    Matrix a;
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            a.mat[i][j] = rand();
        }
    }
    return a;
}

Matrix base;

double benchmark(Matrix (*multiply) (const Matrix&, const Matrix&), Matrix
&result) {
    const int NUM_TEST = 10;
    const int NUM_ITER = 64;
    Matrix a = base;
    result = a;
    double taken = 0;
```

```
    for (int t = 0; t < NUM_TEST; ++t) {
        clock_t start = clock();
        for (int i = 0; i < NUM_ITER; ++i) {
            a = multiply(a, result);
            result = multiply(result, a);
        }
        clock_t finish = clock();
        taken += (double)(finish - start);
    }
    taken /= NUM_TEST;
    printf("Time: %.9f\n", taken / CLOCKS_PER_SEC);
    return taken;
}

int main() {
    base = gen_random_matrix();
    Matrix a, b;
    printf("Slow version\n");
    double slow = benchmark(multiply_naive, a);
    printf("Fast version\n");
    double fast = benchmark(multiply_fast, b);
    if (a == b) {
        printf("Correct answer! Your code is %.2f%% faster\n", slow / fast *
100.0);
    } else {
        printf("Wrong answer!\n");
    }
    return 0;
}
//sok sokong 20211005
```



Phần 3. Bài tập về nhà

Bài tập 11: Tính tích hai đa thức Cho 2 đa thức A(x) và B(x) tương ứng có bậc NN và MM. Hãy tính ma trận tích C(x) = A(x) * B(x) có bậc N+M−1N+M−1. Input: Gồm 2 dòng biểu diễn các đa thức A(x) và B(x), mỗi dòng • Số đầu tiên NN là bậc của đa thức; • N+1N+1 số nguyên tiếp theo, số thứ i là hệ số của xi−1xi−1. Output: Một số nguyên duy nhất là XOR của các hệ số của đa thức C(x). Ví dụ:

Input: 3 83 86 77 15 4 93 35 86 92 49

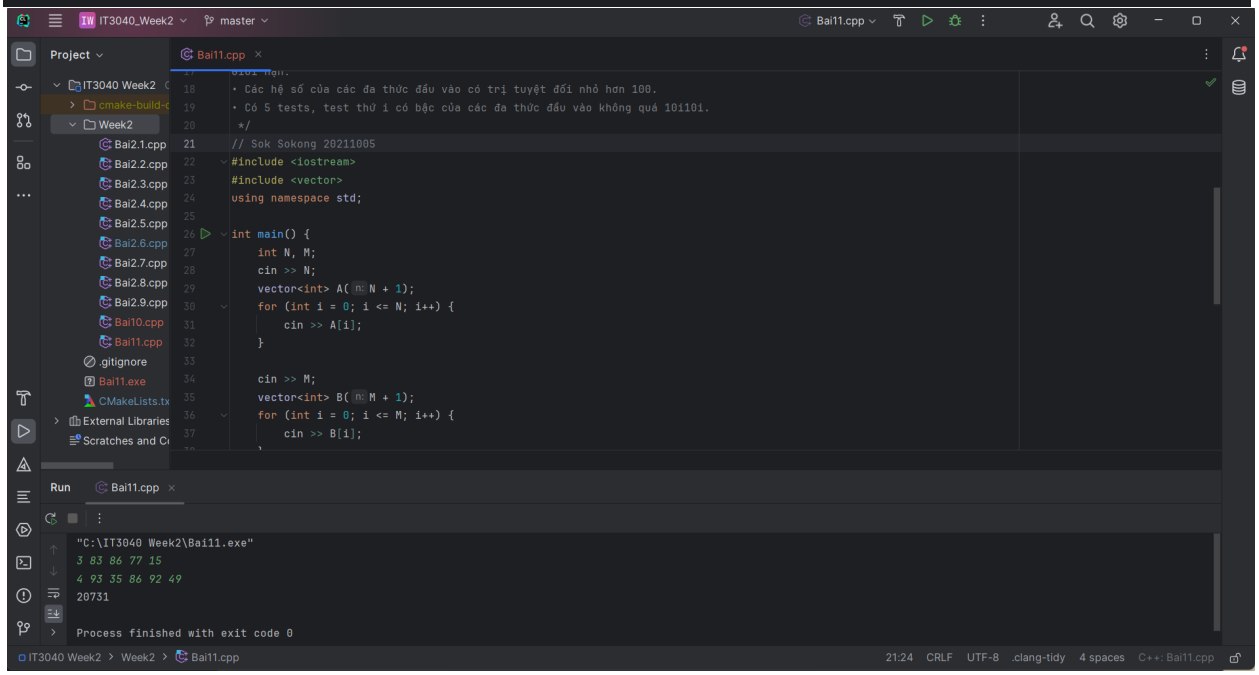
Output: 20731

Giải thích: các hệ số của đa thức kết quả lần lượt là 7719, 10903, 17309, 19122, 19126, 12588, 5153, 735.

Giới hạn: • Các hệ số của các đa thức đầu vào có trị tuyệt đối nhỏ hơn 100. • Có 5 tests, test thứ i có bậc của các đa thức đầu vào không quá 10i10i.

```cpp
// Sok Sokong 20211005
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int N, M;
    cin >> N;
    vector<int> A(N + 1);
    for (int i = 0; i <= N; i++) {
        cin >> A[i];
    }

    cin >> M;
    vector<int> B(M + 1);
    for (int i = 0; i <= M; i++) {
        cin >> B[i];
    }

    int C_size = N + M + 1;
    vector<int> C(C_size, 0);

    for (int i = 0; i <= N; i++) {
        for (int j = 0; j <= M; j++) {
            C[i + j] += A[i] * B[j];
        }
    }

    int result = 0;
    for (int i = 0; i < C_size; i++) {
        result ^= C[i];
    }

    cout << result << endl;
```

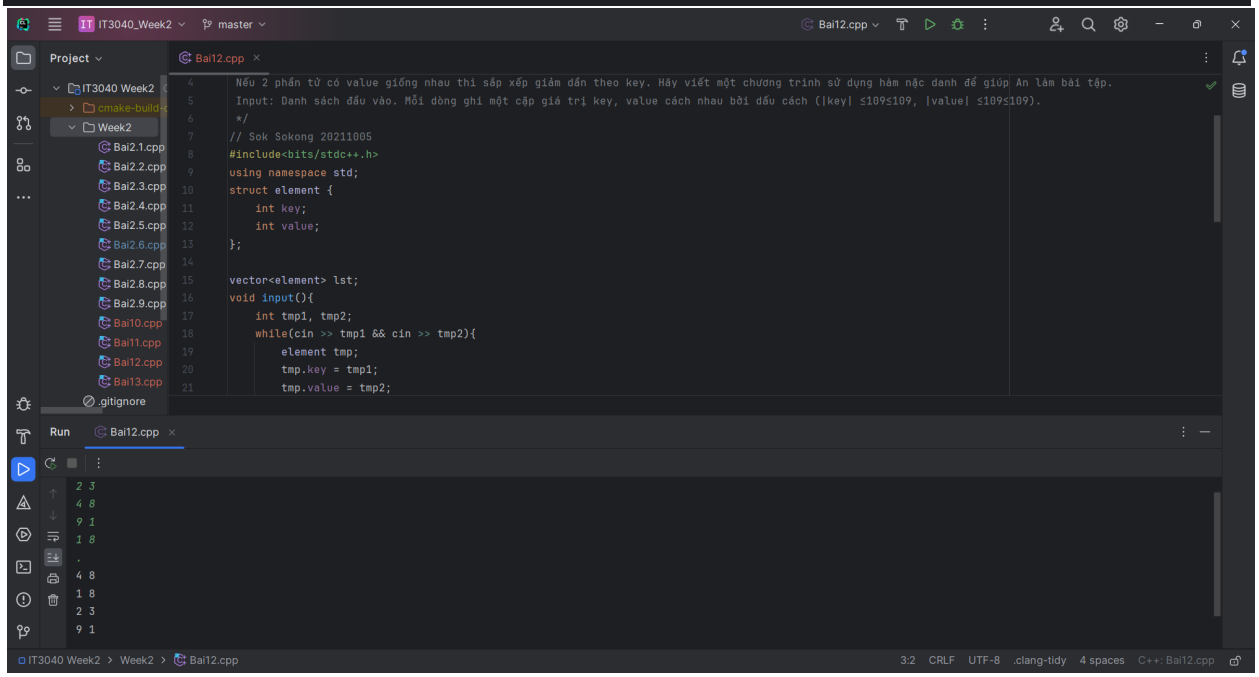
```
    return 0;
}
```



Bài tập 12: Map Sort Hôm nay, cô giáo giao cho An một câu hỏi hóc búa. Cô cho một danh sách với mỗi phần tử có dạng và yêu cầu An sắp xếp danh sách đó giảm dần theo giá trị value. Nếu 2 phần tử có value giống nhau thì sắp xếp giảm dần theo key. Hãy viết một chương trình sử dụng hàm nặc danh để giúp An làm bài tập. Input: Danh sách đầu vào. Mỗi dòng ghi một cặp giá trị key, value cách nhau bởi dấu cách (|key| ≤109≤109, |value| ≤109≤109).

```
// Sok Sokong 20211005
#include<bits/stdc++.h>
using namespace std;
struct element {
   int key;
   int value;
};

vector<element> lst;
void input(){
   int tmp1, tmp2;
   while(cin >> tmp1 && cin >> tmp2){
        element tmp;
        tmp.key = tmp1;
        tmp.value = tmp2;

        lst.push_back(tmp);
   }
}

void print(){
   for(int i=0; i<lst.size(); i++){
```

```
        cout << lst[i].key << " " << lst[i].value << endl;
    }
}

int main(){
    input();
    sort(lst.begin(),lst.end(),[] (element a, element b){
        if(a.value > b.value) return true;
        else if (a.value < b.value) return false;
        else {
            return a.key >= b.key;
        }
    });
    print();
}
```



Bài tập 13: Big Integer

Số nguyên lớn là các số nguyên có giá trị rất lớn và không thể biểu diễn bằng các kiểu dữ liệu nguyên cơ bản. Để biểu diễn số nguyên lớn, ta có thể dùng kiểu struct như sau: struct bigNum{ char sign; char num[101]; };

Nhiệm vụ các bạn là đa năng hóa các toán tử để thực hiện các phép toán số học với kiểu dữ liệu số nguyên lớn vừa định nghĩa ở trên.

```
// Sok Sokong 20211005
#include<bits/stdc++.h>
using namespace std;
struct bigNum {
    char sign;
    char num[101];
};
```

```
// input and preprocess data
void input(bigNum &num1, bigNum &num2){
   string tmp;
   cin >> tmp;
   num1.sign = tmp[0];
   int lens1 = tmp.length() - 1;
   for(int i=0; i<lens1; i++){
       num1.num[100-lens1+i+1] = tmp[i+1];
   }
   for(int i=0; i<100-lens1+1; i++) num1.num[i] = '0';

   cin >> tmp;
   num2.sign = tmp[0];
   int lens2 = tmp.length() - 1;
   for(int i=0; i<lens2; i++){
       num2.num[100-lens2+i+1] = tmp[i+1];
   }
   for(int i=0; i<100-lens2+1; i++) num2.num[i] = '0';
}

// add 2 positive big number
void add(char res[], char *num1, char *num2){
   int c = 0;

   for(int i=100; i>=0; i--){
       int tmp = (int)num1[i] - 48 + (int)num2[i] - 48 + c;
       c = tmp / 10;
       res[i] = tmp % 10 + 48;
   }
}

// sub 2 positive big number, num1 > num2
void sub(char res[], char *num1, char* num2){
   int c = 0;

   for(int i=100; i>=0; i--){
       int tmp1 = (int)num1[i] - 48;
       int tmp2 = (int)num2[i] - 48;

       if(tmp1 >= tmp2 + c){
           res[i] = tmp1 - tmp2 - c + 48;
           c = 0;
       } else {
           tmp1 = tmp1 + 10;
           res[i] = tmp1 - tmp2 - c + 48;
           c = 1;
       }
   }
```

```
}

// multi 2 positive big number
void multi(char res[], char *num1, char *num2){
   // clear array res
   for(int i=0; i<101; i++) res[i] = '0';

   for(int i=100; i>=0; i--){
       // init 1 array temp
       char tmp[101];

       // add i number 0 to last array
       int k;
       for(k = 0; k < i; k++)
           tmp[100-k] = '0';

       int c = 0, sum = 0;
       for(int j=100; j>=0; j--){
           sum = ((int)num1[i] - 48) * ((int)num2[j] - 48) + c;
           tmp[k] = (sum % 10) + 48;
           c = sum / 10;
           k--; if(k < 0) break;
       }

       add(res,tmp,res);
   }
}

// check number1 >= number2
bool check(char *num1, char *num2){
   int foo1, foo2;
   for(foo1 = 0; foo1 < 101; foo1++){
       if(num1[foo1] != '0') break;
   }

   for(foo2 = 0; foo2 < 101; foo2++){
       if(num2[foo2] != '0') break;
   }

   if(foo1 > foo2) return false;
   else if(foo1 < foo2) return true;
   else { // foo1 == foo2
       int foo = foo1;
       while(foo < 101){
           if(num1[foo] < num2[foo]) return false;
           else if (num1[foo] > num2[foo]) return true;
           else {
               foo++;
           }
```

```
        }
    }

    return true;
}

// overloading operator "+"
bigNum operator + (bigNum num1, bigNum num2){
    bigNum res;

    if(num1.sign == '1' && num2.sign == '1'){
        res.sign = '1';
        add(res.num,num1.num,num2.num);
        return res;
    } else if(num1.sign == '1' && num2.sign == '0'){
        if(check(num1.num,num2.num)){
            res.sign = '1';
            sub(res.num,num1.num,num2.num);
            return res;
        } else {
            res.sign = '0';
            sub(res.num,num2.num,num1.num);
            return res;
        }
    } else if(num1.sign == '0' && num2.sign == '1'){
        if(check(num1.num,num2.num)){
            res.sign = '0';
            sub(res.num,num1.num,num2.num);
            return res;
        } else {
            res.sign = '1';
            sub(res.num,num2.num,num1.num);
            return res;
        }
    } else {
        res.sign = '0';
        add(res.num,num1.num,num2.num);
        return res;
    }
}

bigNum operator - (bigNum num1, bigNum num2){
    bigNum res;

    if(num1.sign == '1' && num2.sign == '0'){
        num2.sign = '1';
        res = num1 + num2;
        return res;
    } else if(num1.sign == '1' && num2.sign == '1'){
```

```
        num2.sign = '0';
        res = num1 + num2;
        return res;
    } else if(num1.sign == '0' && num2.sign == '1'){
        num2.sign = '0';
        res = num1 + num2;
        return res;
    } else {
        num2.sign = '1';
        res = num1 + num2;
        return res;
    }
}

bigNum operator * (bigNum num1, bigNum num2){
    bigNum res;

    if(num1.sign == '1' && num2.sign == '1'){
        res.sign = '1';
        multi(res.num,num1.num,num2.num);
        return res;
    } else if(num1.sign == '1' && num2.sign == '0'){
        res.sign = '0';
        multi(res.num,num1.num,num2.num);
        return res;
    } else if(num1.sign == '0' && num2.sign == '1'){
        res.sign = '0';
        multi(res.num,num1.num,num2.num);
        return res;
    } else {
        res.sign = '1';
        multi(res.num,num1.num,num2.num);
        return res;
    }
}

// print bignumber
void printBigNumber(bigNum number){
    cout << number.sign;
    int start;
    for(start=0; start<101; start++)
        if(number.num[start] != '0') break;

    for(int i = start; i<101; i++)
        cout << number.num[i];
}

int main(){
    bigNum num1, num2;
```

```
    input(num1,num2);

    bigNum so3, so4;
    so3.sign = '1', so4.sign = '1';
    for(int i=0; i<100; i++){
        so3.num[i] = '0';
        so4.num[i] = '0';
    }
    so3.num[100] = 3 + 48;
    so4.num[100] = 4 + 48;

    bigNum res = num1*num2 - so3 * num1 + so4 * num2;

    printBigNumber(res);
}
```