Mục lục nội dung

[Bài thực hành số 2 – Tuần 10 3](#_Toc161238431)

[BÀI TẬP TRÊN LAP 3](#_Toc161238432)

**Phần 1. Thực hành về hàm**

**Truyền tham trị, tham chiếu và tham số ngầm định**

[**Bài tập 2.1.**Truyền tham trị 3](#_Toc161238433)

[**Bài tập 2.2.**Truyền tham chiếu. 5](#_Toc161238434)

[**Bài tập 2.3.** Tham số ngầm định. 7](#_Toc161238435)

**Đa năng hóa hàm**

[**Bài tập 2.4.** Đa năng hóa hàm. 9](#_Toc161238436)

[**Bài tập 2.5.** Đa năng hóa toán tử. 11](#_Toc161238437)

**Con trỏ hàm và tham số hóa hàm**

[**Bài tập 2.6.** Con trỏ hàm 15](#_Toc161238438)

[**Bài tập 2.7.** Khái quát hóa hàm. 18](#_Toc161238439)

**Biểu thức lamda và hàm nặc danh**

[**Bài tập 2.8.** Sắp xếp 21](#_Toc161238440)

**Phần 2. Thực hành về tối ưu mã nguồn**

[**Bài tập 2.9.**Tính hàm sigmoid 24](#_Toc161238433)

[**Bài tập 2.10.** (bonus): Tính tích hai ma trận vuông. 5](#_Toc161238434)

[**BÀI TẬP VỀ NHÀ** 8](#_Toc161238441)

[**Bài tập 2.11.** Tính tích hai đa thức 8](#_Toc161238442)

[**Bài tập 2.12.** Map Sort 8](#_Toc161238443)

[**Bài tập 2.13.** Big Integer 8](#_Toc161238443)

Mục lục hình ảnh

[Hình 1 Bài 2.1 Viết hàm tính độ dài cạnh huyền của tam giác 3](#_Toc161238452)

[Hình 2 Bài 2.2 Viết hoán vị vòng tròn 3 biến a,b,c 5](#_Toc161238453)

[Hình 3 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. 7](#_Toc161238454)

[Hình 4 Bài 2.4 Viết hàm tính lập phương số nguyên và số thực 9](#_Toc161238455)

[Hình 5 Bài 2.5 Viết các toán tử tính tổng hiệu, tích và thương của 2 số phức 11](#_Toc161238456)

[Hình 6 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.](#_Toc161238457)

[Viết chương trình mô phỏng lại quá trình biến đổi để kiếm chứng giả thuyết với giá trị của n nhập từ bàn phím 15](#_Toc161238457)

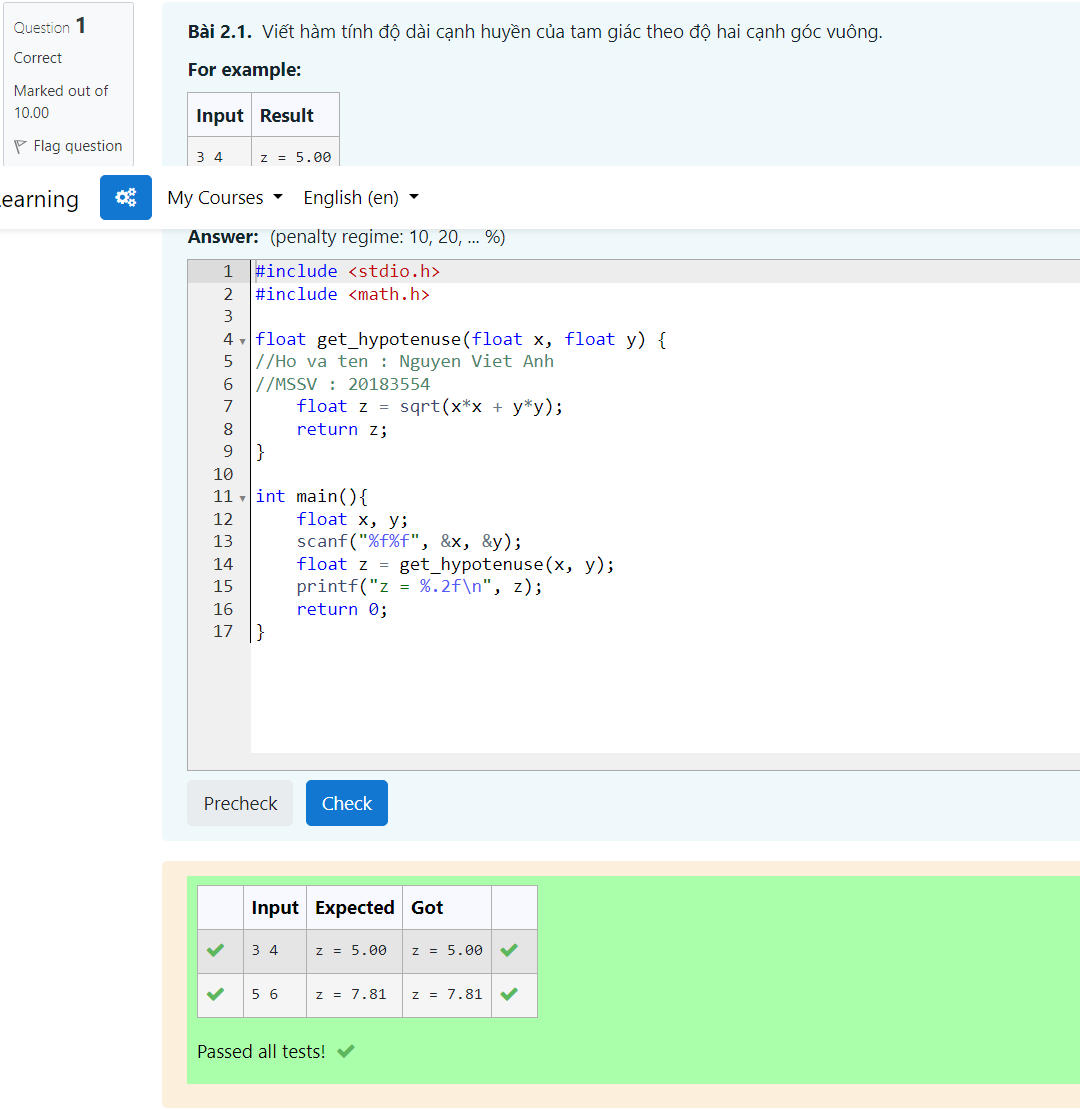
[Hình 7 Bài 2.7 Viết hàm tính tổng các phần tử trong 2 mảng 18](#_Toc161238458)

[Hình 8 Bài 2.8 Viết hàm so sánh thuật toán sắp xếp 21](#_Toc161238459)

[Hình 9 Bài 2.9 Tính hàm sigmoid 24](#_Toc161238458)

# Bài thực hành số 2 – Tuần 11

# BÀI TẬP TRÊN LAP

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

Hình Bài 2.1: Viết hàm tính độ dài cạnh huyền của tam giác

#include <stdio.h>

#include <math.h>

float get\_hypotenuse(float x, float y) {

//Ho va ten : Nguyen Viet Anh

//MSSV : 20183554

float z = sqrt(x\*x + y\*y);

return z;

}

int main(){

float x, y;

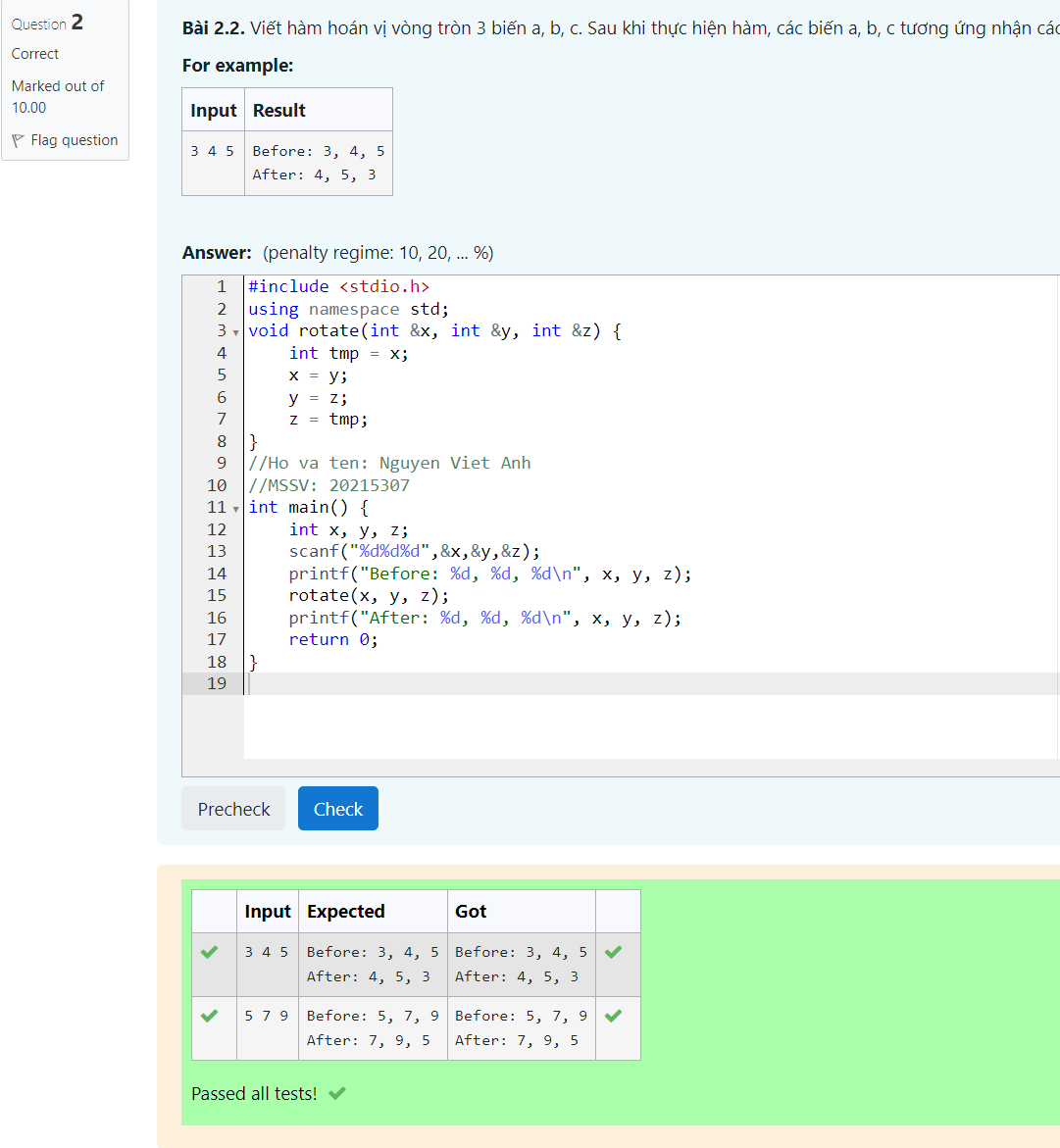
scanf("%f%f", &x, &y);

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. 

Hình Bài 2.2: Viết hoán vị vòng tròn 3 biến a,b,c

#include <stdio.h>

using namespace std;

void rotate(int &x, int &y, int &z) {

int tmp = x;

x = y;

y = z;

z = tmp;

}

//Ho va ten: Nguyen Viet Anh

//MSSV: 20215307

int main() {

printf("Ho va ten: Nguyen Viet Anh\n");

printf("MSSV:20215307\n");

int x, y, z;

scanf("%d%d%d",&x,&y,&z);

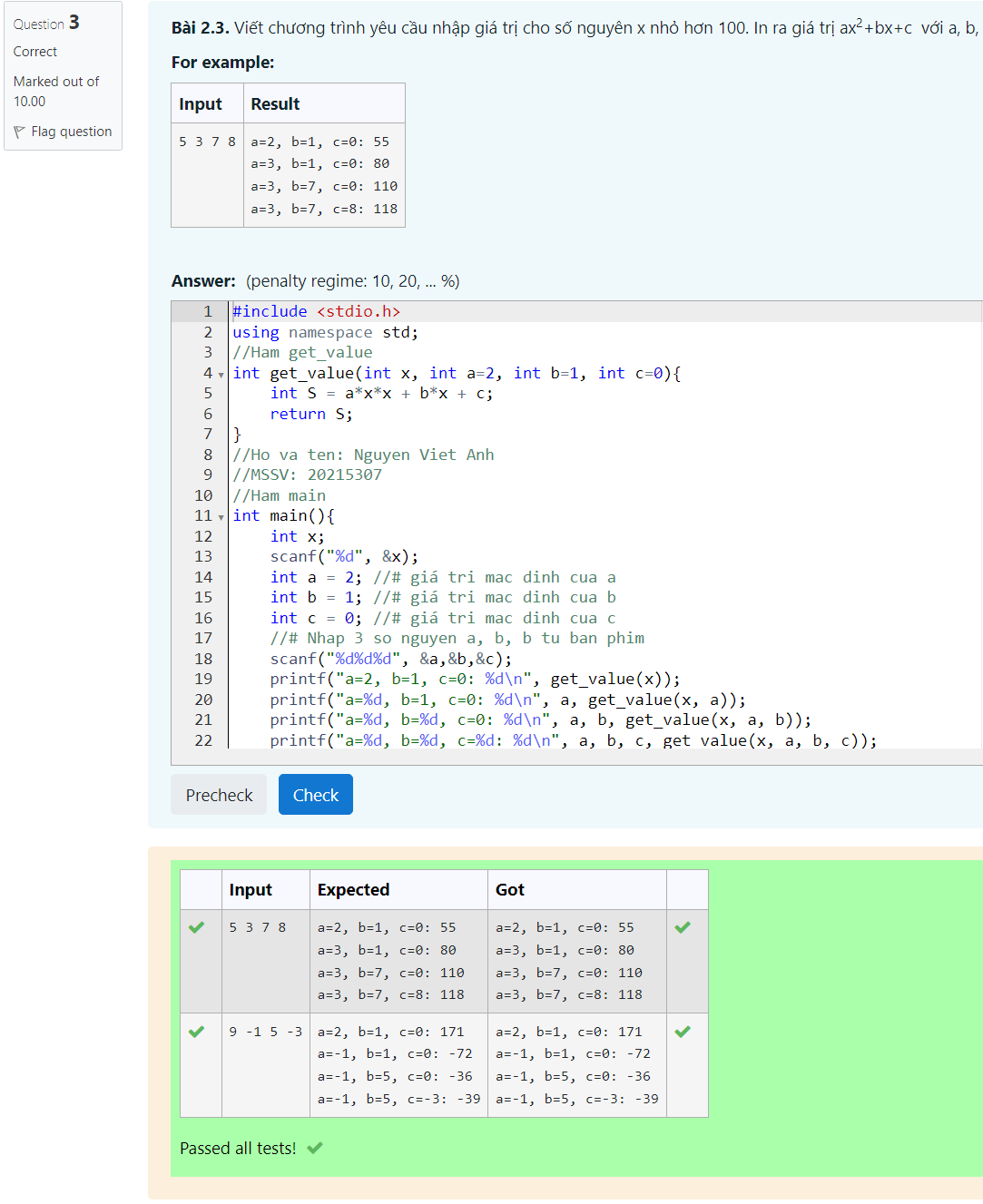
printf("Before: %d, %d, %d\n", x, y, z);

rotate(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. 

Hình 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>

using namespace std;

//Ham get\_value

int get\_value(int x, int a=2, int b=1, int c=0){

int S = a\*x\*x + b\*x + c;

return S;

}

//Ho va ten: Nguyen Viet Anh

//MSSV: 20215307

//Ham main

int main(){

printf("Ho Va Ten: Nguyen Viet Anh\n");

printf("MSSV: 20215307\n");

int x;

scanf("%d", &x);

int a = 2; //# giá tri mac dinh cua a

int b = 1; //# giá tri mac dinh cua b

int c = 0; //# giá tri mac dinh cua c

//# Nhap 3 so nguyen a, b, b tu ban phim

scanf("%d%d%d", &a,&b,&c);

printf("a=2, b=1, c=0: %d\n", get\_value(x));

printf("a=%d, b=1, c=0: %d\n", a, get\_value(x, a));

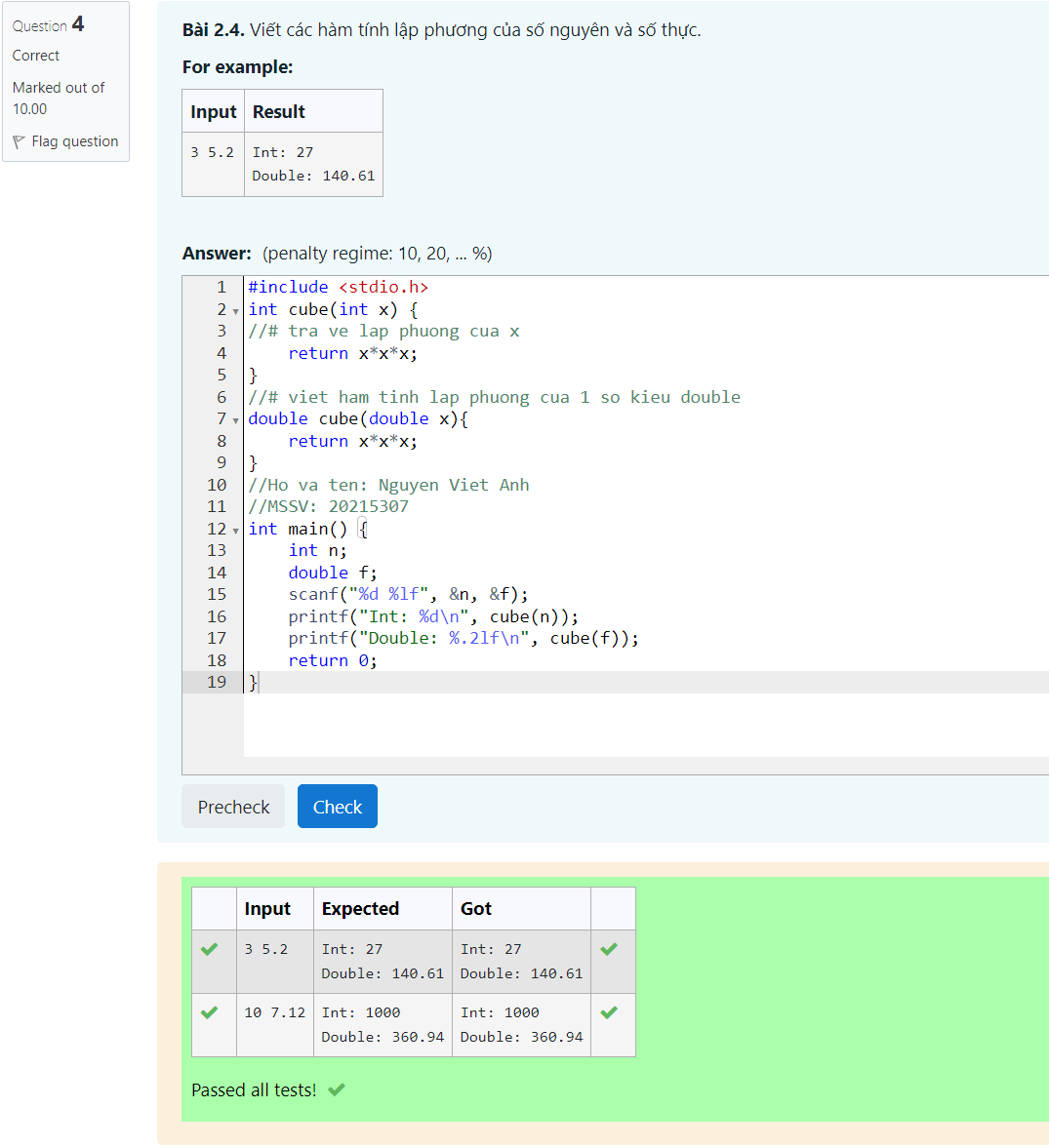
printf("a=%d, b=%d, c=0: %d\n", a, b, get\_value(x, a, b));

printf("a=%d, b=%d, c=%d: %d\n", a, b, c, get\_value(x, a, b, c));

return 0;

}

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



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

#include <stdio.h>

int cube(int x) {

//# tra ve lap phuong cua x

return x\*x\*x;

}

//# viet ham tinh lap phuong cua 1 so kieu double

double cube(double x){

return x\*x\*x;

}

//Ho va ten: Nguyen Viet Anh

//MSSV: 20215307

int main() {

printf("Ho Va Ten: Nguyen Viet Anh\n");

printf("MSSV: 20215307\n");

int n;

double f;

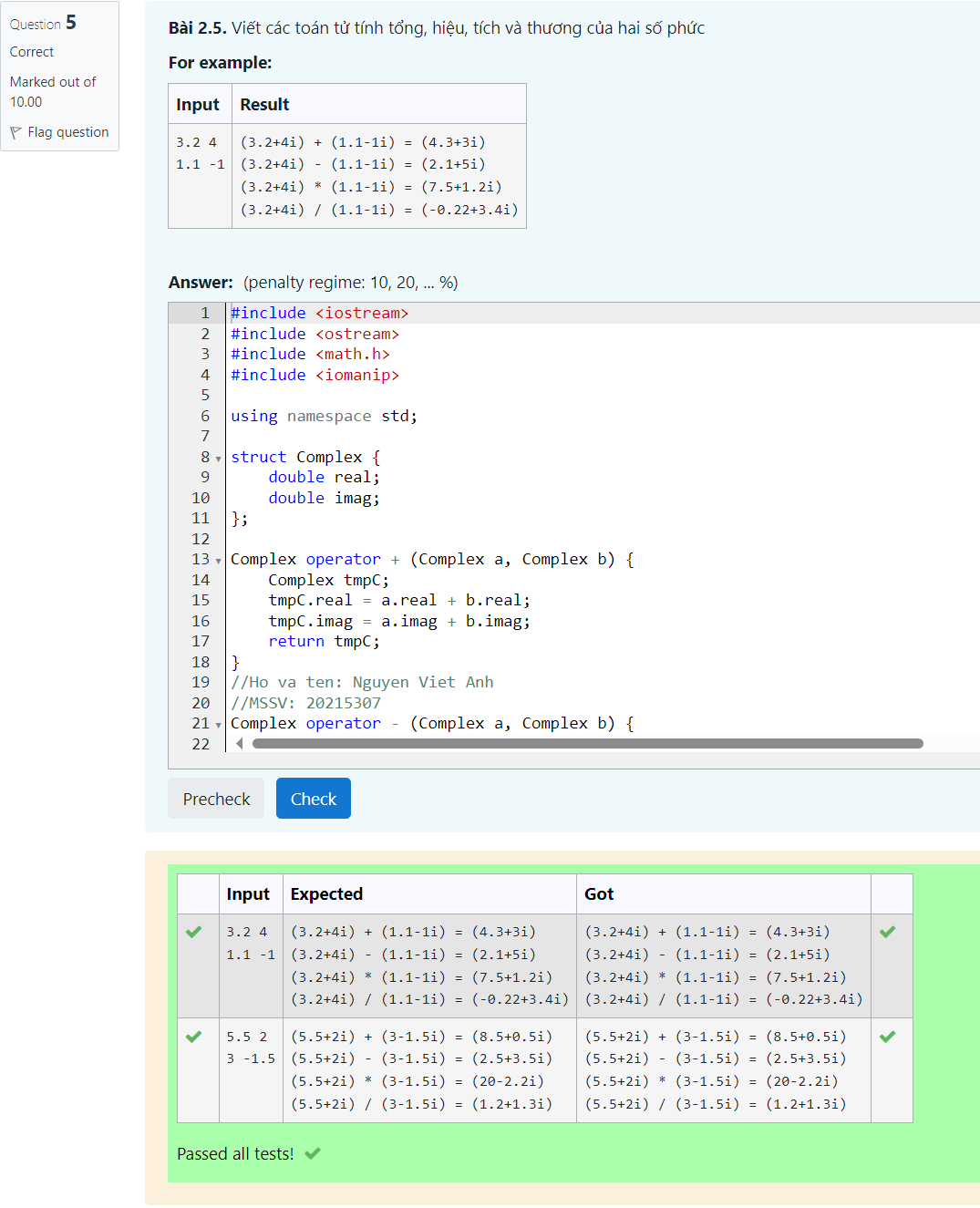
scanf("%d %lf", &n, &f);

printf("Int: %d\n", cube(n));

printf("Double: %.2lf\n", cube(f));

return 0;

}

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

Hình Bài 2.5 Viết các toán tử tính tổng, hiệu, tích, thương của 2 số phức

#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;

}

//Ho va ten: Nguyen Viet Anh

//MSSV: 20215307

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;

}

//Ho va ten: Nguyen Viet Anh

//MSSV: 20215307

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() {

printf("Ho Va Ten: Nguyen Viet Anh\n");

printf("MSSV:20215307\n");

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;

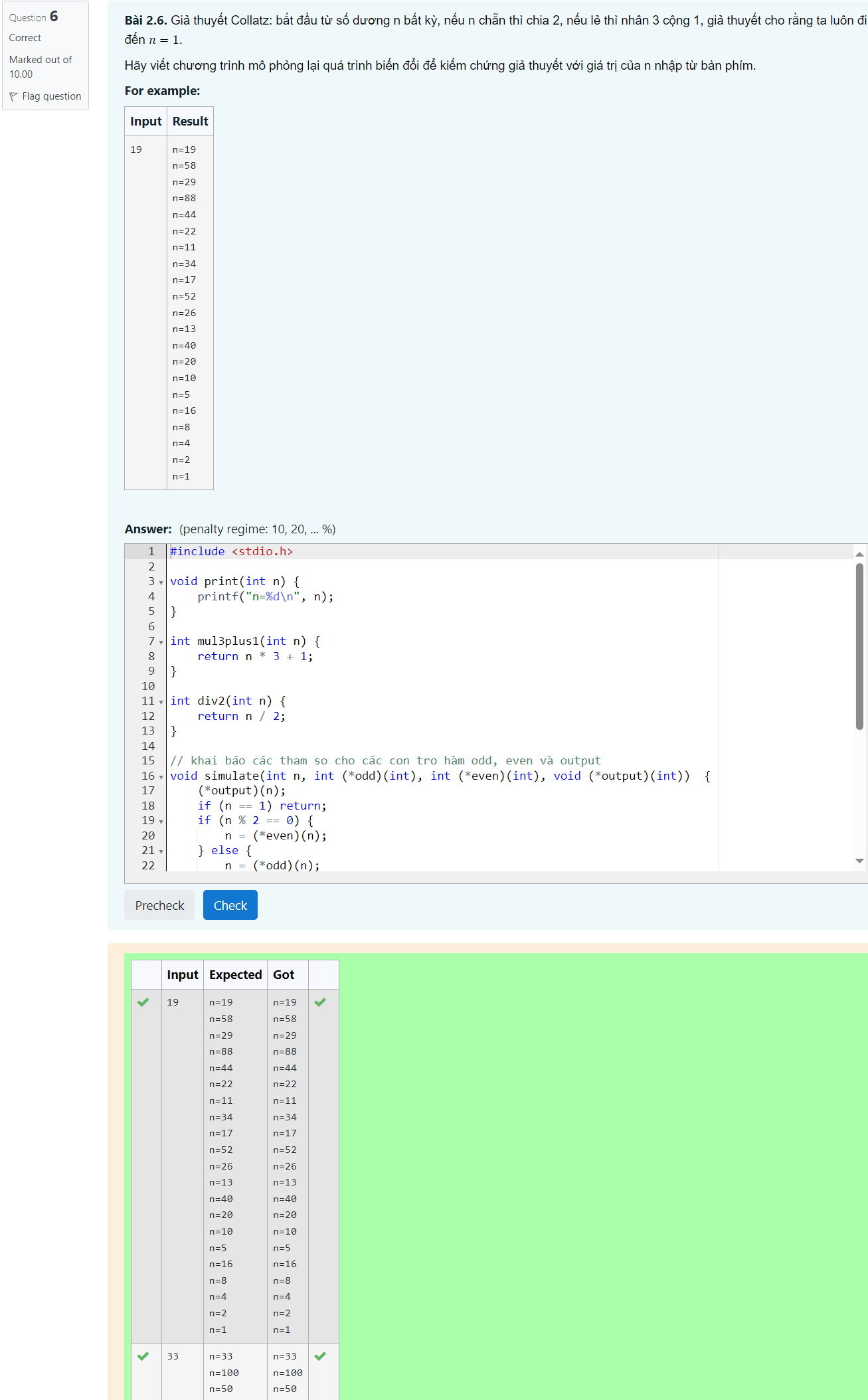
cout << a << " \* " << b << " = " << a \* b << endl;

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 kiếm chứng giả thuyết với giá trị của n nhập từ bàn phím.



Hình Bài 2.6 Viết chương trình mô phỏng lại quá trình biến đổi để kiểm chứng giả thuyết Collatz

#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;

}

// khai báo các tham so cho các con tro hàm odd, even và output

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);

}

//Ho va ten: Nguyen Viet Anh

//MSSV: 20215307

int main() {

printf("Ho Va Ten: Nguyen Viet Anh\n");

printf("MSSV: 20215307\n");

int (\*odd)(int) = NULL;

int (\*even)(int) = NULL;

odd = mul3plus1;

even = div2;

int n;

scanf("%d", &n);

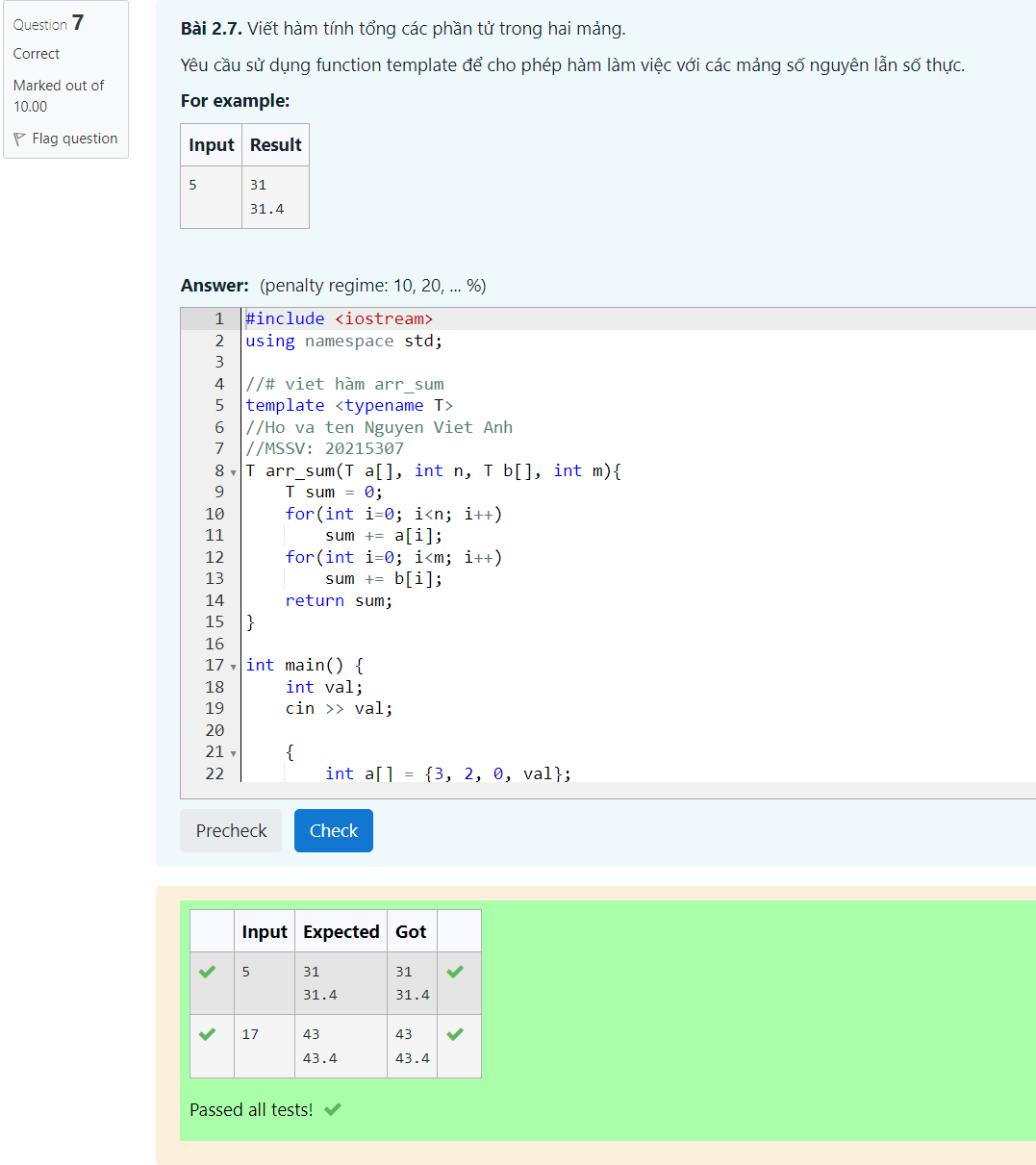
simulate(n, odd, even, print);

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.



Hình Bài 2.7 Viết hàm tính tổng các phần tử trong 2 mảng

#include <iostream>

using namespace std;

//# viet hàm arr\_sum

template <typename T>

//Ho va ten Nguyen Viet Anh

//MSSV: 20215307

T arr\_sum(T a[], int n, T b[], int m){

T sum =0;

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

sum += a[i];

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

sum += b[i];

return sum;

}

int main() {

printf("Ho Va Ten: Nguyen Viet Anh\n");

printf("MSSV: 20215307\n");

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};

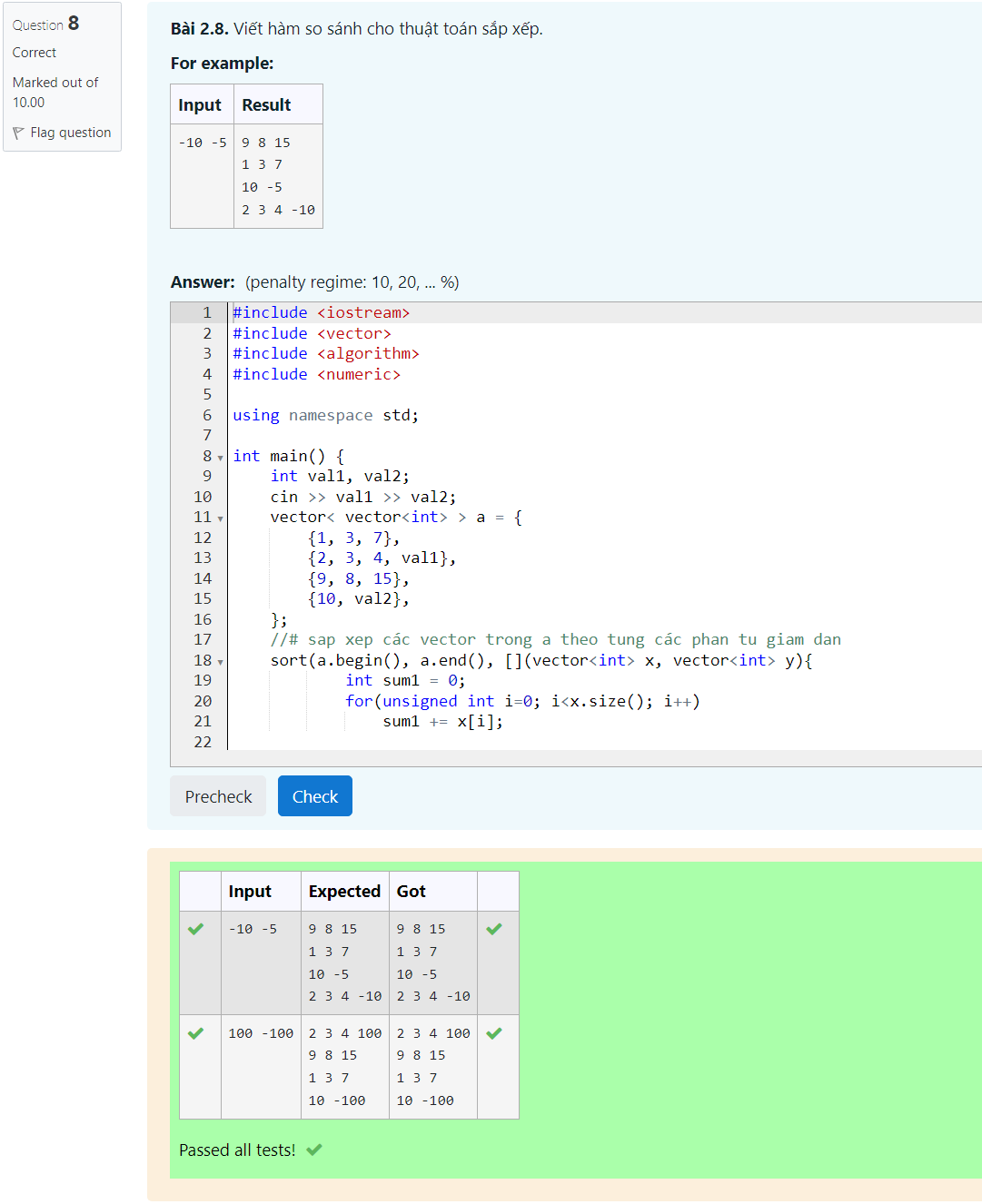
double b[] = {5, 6.1, 1, 2.3, 7};

cout << arr\_sum(a, 4, b, 5) << endl;

}

return 0;

}

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

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

#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

using namespace std;

int main() {

printf("Ho Va Ten: Nguyen Viet Anh\n");

printf("MSSV:20215307\n");

int val1, val2;

cin >> val1 >> val2;

vector< vector<int> > a = {

{1, 3, 7},

{2, 3, 4, val1},

{9, 8, 15},

{10, val2},

};

//# sap xep các vector trong a theo tung các phan tu giam dan

sort(a.begin(), a.end(), [](vector<int> x, vector<int> y){

int sum1 = 0;

for(unsigned int i=0; i<x.size(); i++)

sum1 += x[i];

int sum2 = 0;

for(unsigned int i=0; i<y.size(); i++)

sum2 += y[i];

return sum1 > sum2;

});

//Ho va ten: Nguyen Viet Anh

//MSSV 20215307

for (const auto &v : a) {

for (int it : v) {

cout << it << ' ';

}

cout << endl;

}

return 0;

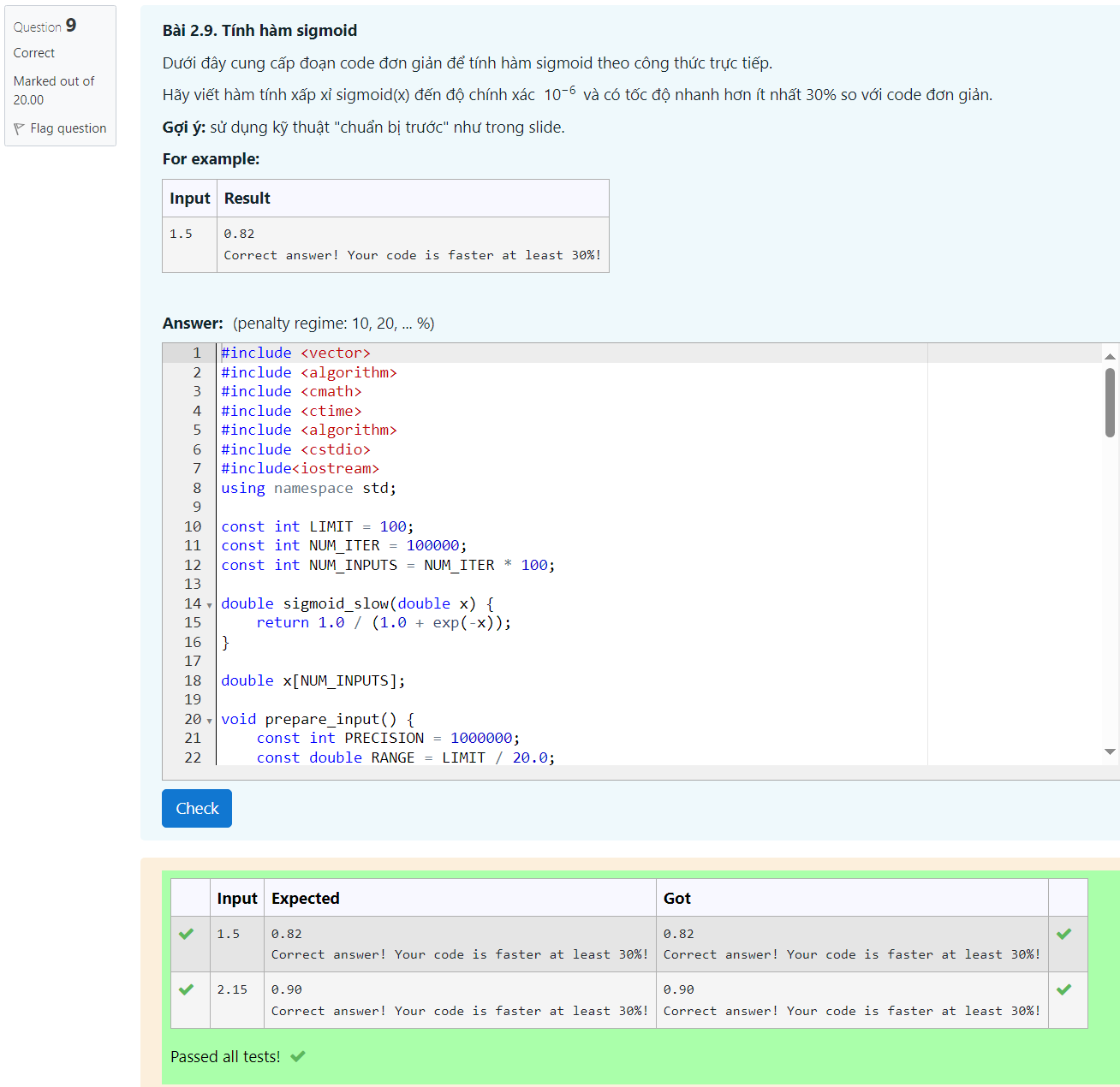
}

**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.



*Hình 9 Bài 2.9 Tính hàm sigmoid*

#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;

}

}

//# BEGIN fast code

//# khai báo các bien phu tro can thiet

#define MAX\_N 100000

#define denta 0.0001

double sigmoid[MAX\_N];

const double start = -5.0;

const double stop = 5.0;

//# hàm chuan bi du lieu

void precalc() {

double foo = start;

for(int i=0; i<MAX\_N; i++){

sigmoid[i] = sigmoid\_slow(foo);

foo += denta;

}

}

//Ho va ten Nguyen Viet Anh

//MSSV : 20215307

//# hàm tính sigmoid(x) nhanh sigmoid\_fast(x)

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);

}

//# END fast code

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);

//# printf("Time: %.9f\n", taken / CLOCKS\_PER\_SEC);

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() {

printf("Ho Va Ten: NguyenViet Anh\n");

printf("MSSV: 20215307\n");

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("Wrong answer or your code is not fast enough!\n");

printf("Correct answer! Your code is faster at least 30%%!\n");

}

return 0;

}

# **BÀI TẬP VỀ NHÀ**

Chụp ảnh kết quả của tất cả các test.

**Bài 2.11: 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).

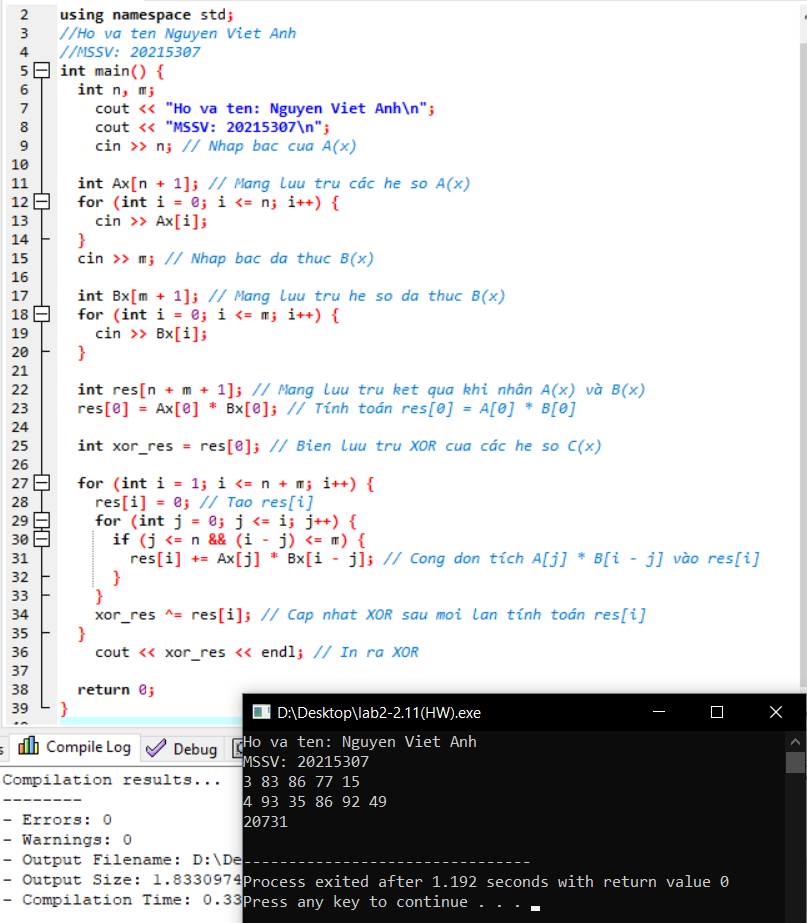
Ví dụ: Input: 3 83 86 77 15 4 93 35 86 92 49 Output: 20731

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.



#include <iostream>

using namespace std;

//Ho va ten Nguyen Viet Anh

//MSSV: 20215307

int main() {

int n, m;

printf("Ho va ten: Nguyen Viet Anh\n");

printf ("MSSV: 20215307\n");

cin >> n; // Nhap bac cua A(x)

int Ax[n + 1]; // Mang luu tru các he so A(x)

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

cin >> Ax[i];

}

cin >> m; // Nhap bac da thuc B(x)

int Bx[m + 1]; // Mang luu tru he so da thuc B(x)

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

cin >> Bx[i];

}

int res[n + m + 1]; // Mang luu tru ket qua khi nhân A(x) và B(x)

res[0] = Ax[0] \* Bx[0]; // Tính toán res[0] = A[0] \* B[0]

int xor\_res = res[0]; // Bien luu tru XOR cua các he so C(x)

for (int i = 1; i <= n + m; i++) {

res[i] = 0; // Tao res[i]

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

if (j <= n && (i - j) <= m) {

res[i] += Ax[j] \* Bx[i - j]; // Cong don tích A[j] \* B[i - j] vào res[i]

}

}

xor\_res ^= res[i]; // Cap nhat XOR sau moi lan tính toán res[i]

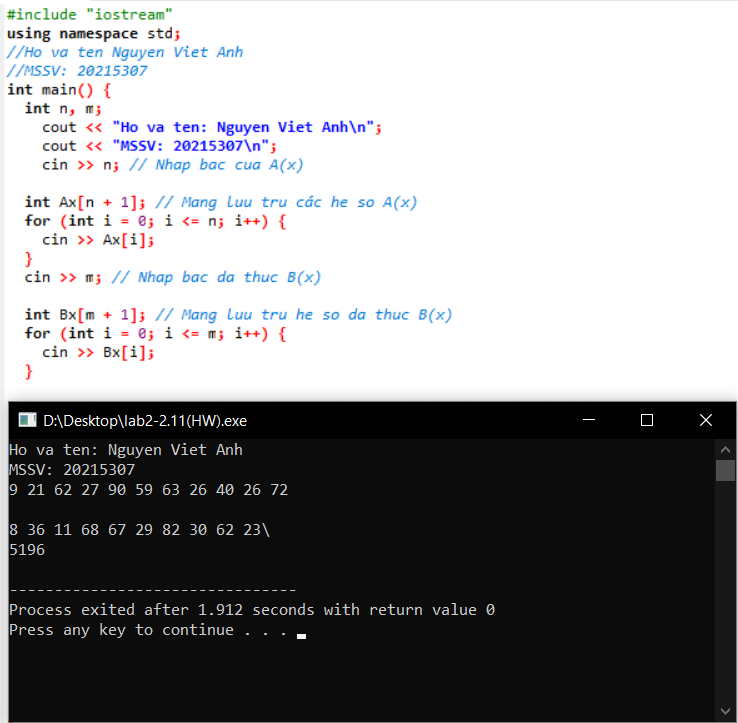
}

cout << xor\_res << endl; // In ra XOR

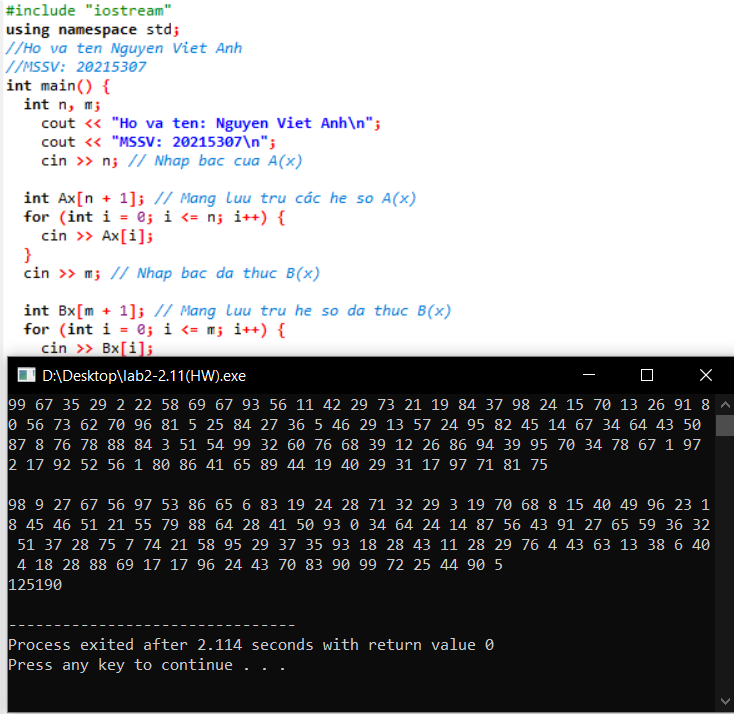
return 0;

}

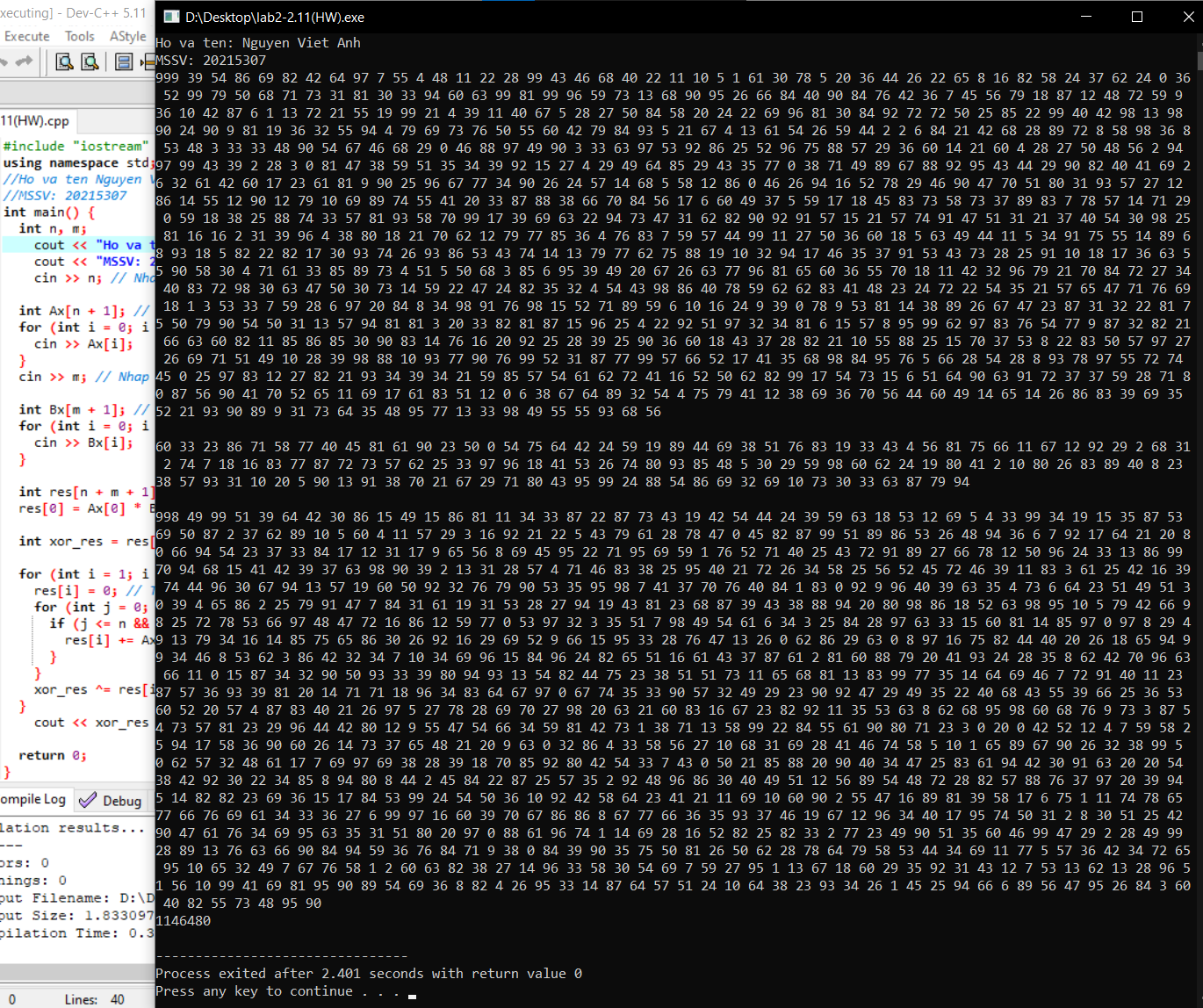
**Case 1**

****

**Case 2**

****

**Case 3**



**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).

Output: In danh sách đã được sắp xếp theo yêu cầu. Mỗi dòng ghi một cặp giá trị key, value cách nhau bởi dấu cách.

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

//Cau truc T gom cap khoa va gia tri

struct T{

int key;

int value;

};

//Ho va ten Nguyen Viet Anh

//MSSV: 20215307

vector<T> list; //Vector list luu tru các phan tu co kieu du lieu la T

int main(){

printf("Ho va ten: Nguyen Viet Anh\n");

printf ("MSSV: 20215307\n");

int k, v;

while(cin >> k >> v){ //Doc gia tri vao k va v cho den khi khong hop le hoac gap so am

if(k <0 || v < 0) break;

T tmp; //bien cau truc tam thoi kieu T

tmp.key = k;

tmp.value = v;

list.push\_back(tmp); //Day vao vector list

}

sort(list.begin(), list.end(), [](T a, T b){

if(a.value > b.value ) return true; //Sap xep theo value giam dan

else if(a.value < b.value ) return false;

else return a.key >= b.key;

});

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

cout << list[i].key << " " << list[i].value << endl;

}

return 0;

}

**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;

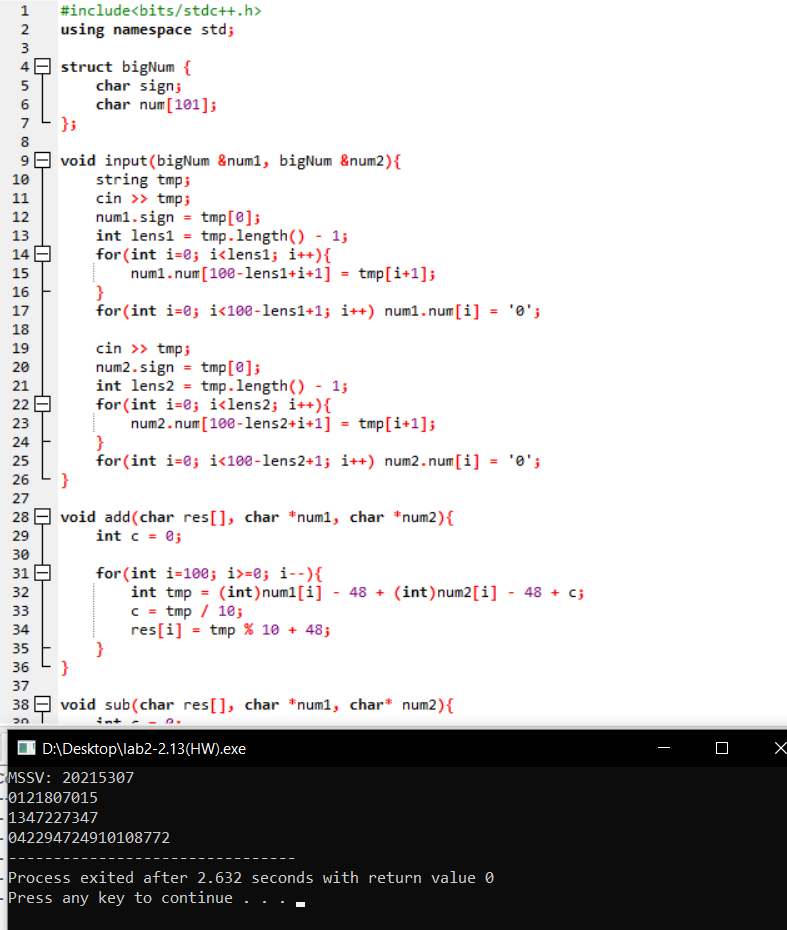
char num[101];

};

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.

Input: Dữ liệu vào gồm hai dòng mô tả hai số nguyên lớn aa và bb, mỗi dòng chứa 1 chuỗi ký tự mô tả 1 số nguyên lớn không vượt quá 1010010100. Chữ số đầu của mỗi chuỗi ký tự sẽ thể hiện dấu của số đó: 0 là âm, 1 là dương. Các chữ số sau thể hiện giá trị của số đó.

Output: In ra giá trị của biểu thức ab−3a+4bab−3a+4b. Kết quả in ra một số nguyên lớn dưới dạng chuỗi ký tự có định dạng như mô tả trong dữ liệu vào.

****

#include<bits/stdc++.h>

using namespace std;

struct bigNum {

char sign;

char num[101];

};

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';

}

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;

}

}

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;

}

}

}

void multi(char res[], char \*num1, char \*num2){

for(int i=0; i<101; i++) res[i] = '0';

for(int i=100; i>=0; i--){

char tmp[101];

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);

}

}

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 {

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;

}

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;

}

}

//Ho va ten: Nguyen Viet Anh

//MSSV:20215307

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(){

printf("Ho va ten: Nguyen Viet Anh\n");

printf("MSSV: 20215307\n");

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;

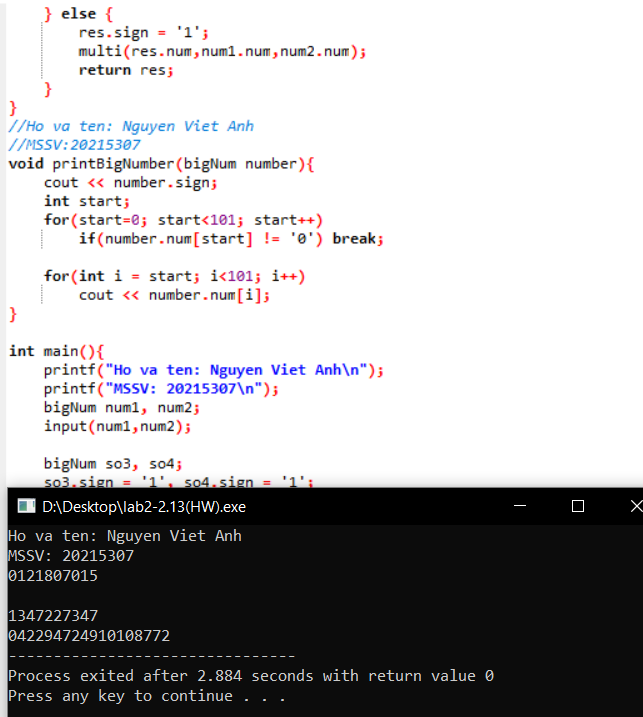
so4.num[100] = 4 + 48;

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

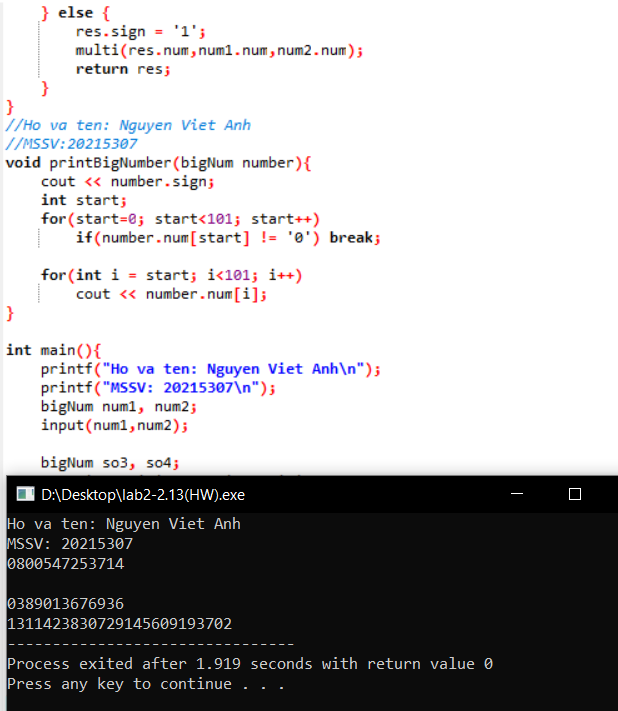
printBigNumber(res);

}

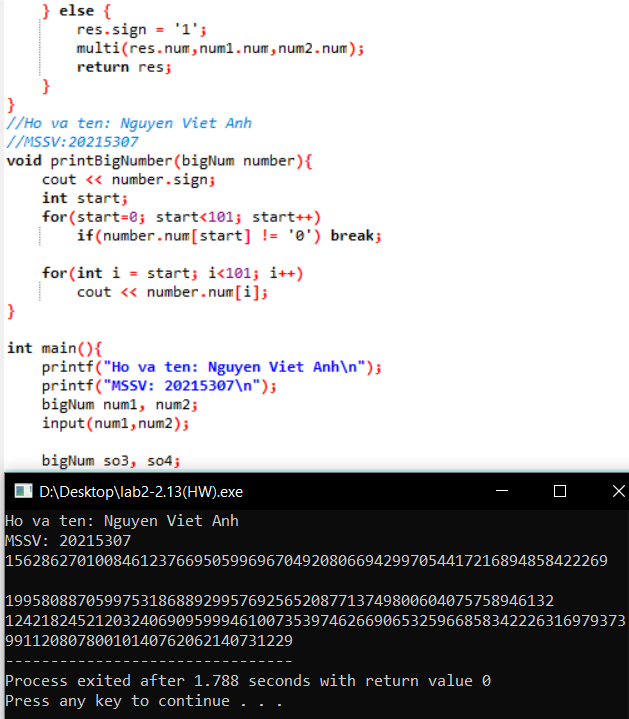
**Case 1**

****

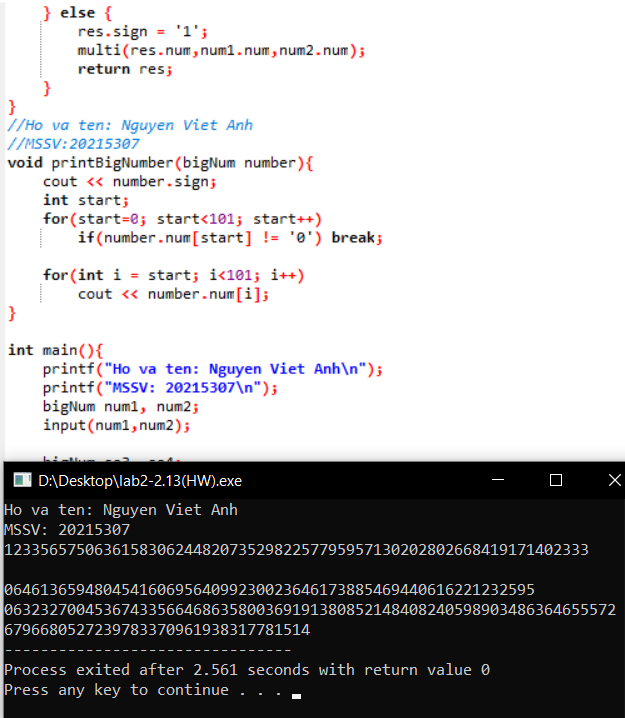
**Case 2**

****

**Case 3**

****

**Case 4**

****