Mục Lục

[Bài thực hành số 2 – Tuần 10 2](#_Toc165752990)

[Phần 1. Thực hành về hàm 2](#_Toc165752991)

[1.1 Truyền tham trị, tham chiếu và tham số ngầm định 2](#_Toc165752992)

[Bài tập 1: Truyền tham trị 2](#_Toc165752993)

[Bài tập 2: Truyền tham chiếu 3](#_Toc165752994)

[Bài tập 3: Tham số ngầm định 5](#_Toc165752995)

[1.2 Đa năng hóa hàm 7](#_Toc165752996)

[Bài tập 4: Đa năng hóa hàm 7](#_Toc165752997)

[Bài tập 5: Đa năng hóa toán tử 8](#_Toc165752998)

[1.3 Con trỏ hàm và tham số hóa hàm 12](#_Toc165752999)

[Bài tập 6: Con trỏ hàm 12](#_Toc165753000)

[Bài tập 7: Khái quát hóa hàm 16](#_Toc165753001)

[1.4 Biểu thức lamda và hàm nặc danh 19](#_Toc165753002)

[Bài tập 8: Sắp xếp 19](#_Toc165753003)

[Phần 2. Thực hành về tối ưu mã nguồn 21](#_Toc165753004)

[Bài tập 9: Tính hàm sigmoid 21](#_Toc165753005)

[Bài tập 10 (bonus): Tính tích hai ma trận vuông 25](#_Toc165753006)

[Phần 3. Bài tập về nhà 30](#_Toc165753007)

[Bài tập 11: Tính tích hai đa thức 30](#_Toc165753008)

[Bài tập 12: Map Sort 33](#_Toc165753009)

[Bài tập 13: Big Integer 37](#_Toc165753010)

# Bài thực hành số 2 – Tuần 10

# Phần 1. Thực hành về hàm

## 1.1 Truyền tham trị, tham chiếu và tham số ngầm định

### Bài tập 1: Truyền tham trị

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

//Cau 1

// Nhữ Ngọc Minh 20225046

#include <stdio.h>

#include <math.h>

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

return sqrt(x\*x+y\*y);

}

int main()

{ float x, y;

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

float z = get\_hypotenuse(x, y);

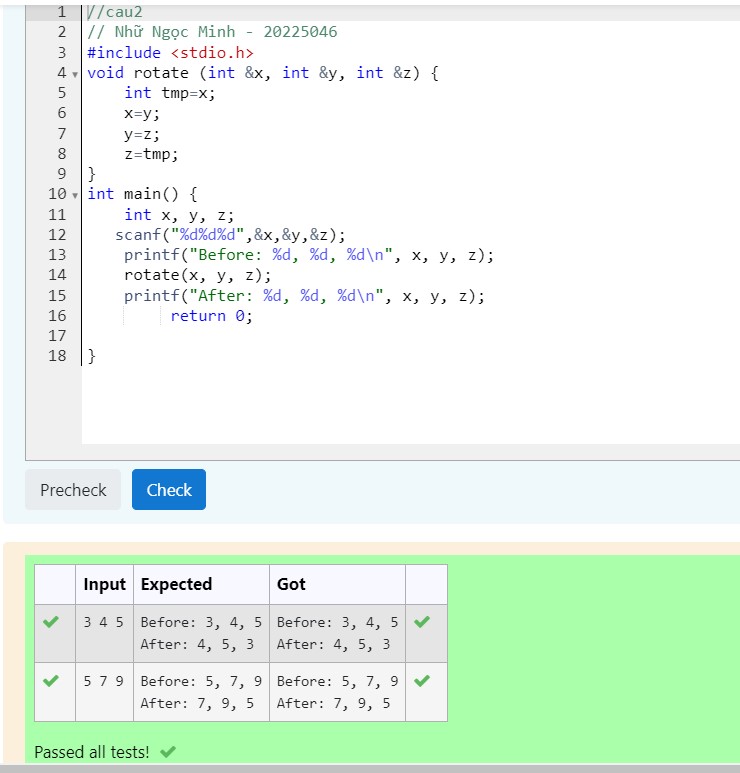
printf("z = %.2f\n", z);

return 0;

}

### Bài tập 2: Truyền tham chiếu

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.



//cau2

// Nhữ Ngọc Minh - 20225046

#include <stdio.h>

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

int tmp=x;

x=y;

y=z;

z=tmp;

}

int main() {

int x, y, z;

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

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

rotate(x, y, z);

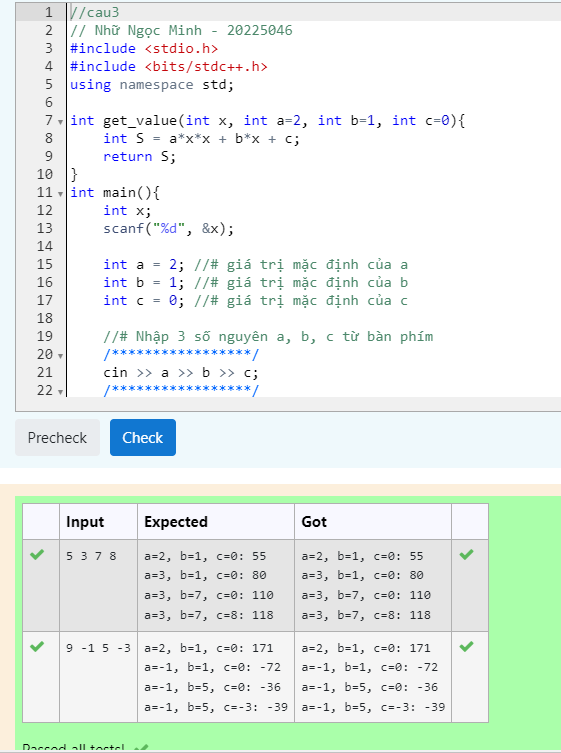
printf("After: %d, %d, %d\n", x, y, z);

return 0;

}

### Bài tập 3: Tham số ngầm định

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



//cau3

// Nhữ Ngọc Minh - 20225046

#include <stdio.h>

#include <bits/stdc++.h>

using namespace std;

int get\_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; //# giá trị mặc định của a

int b = 1; //# giá trị mặc định của b

int c = 0; //# giá trị mặc định của c

//# Nhập 3 số nguyên a, b, c từ bàn phím

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

cin >> 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));

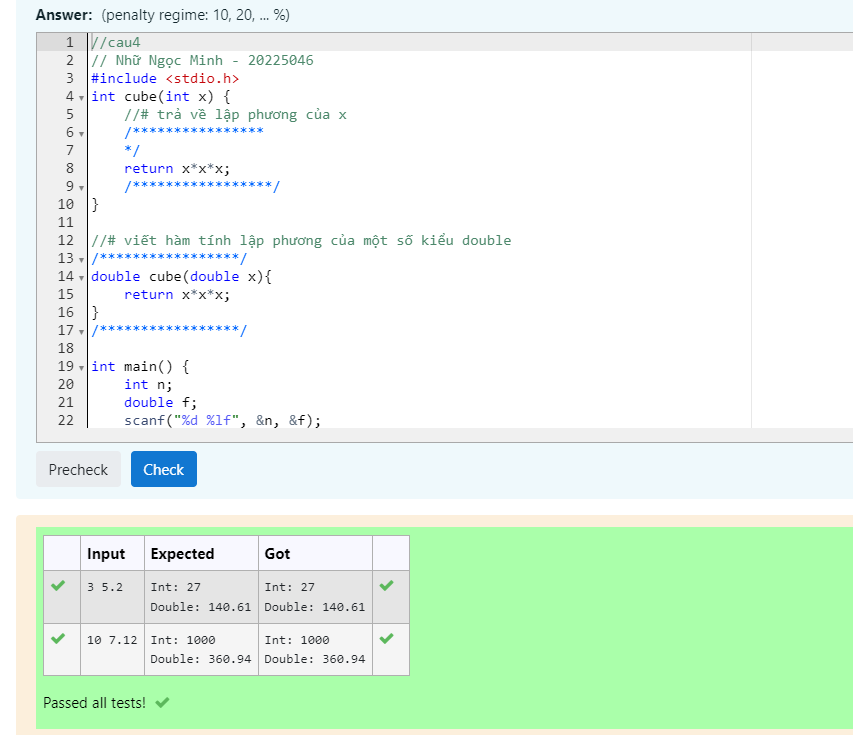
return 0;

}

## 1.2 Đa năng hóa hàm

### Bài tập 4: Đa năng hóa hàm

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



//cau4  
// Nhữ Ngọc Minh - 20225046  
#include <stdio.h>  
int cube(int x) {  
    //# trả về lập phương của x  
    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
    \*/  
    return x\*x\*x;  
    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
}

//# viết hàm tính lập phương của một số kiểu double  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
double cube(double x){  
    return x\*x\*x;  
}  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

    printf("Int: %d\n", cube(n));  
    printf("Double: %.2lf\n", cube(f)); // in ra mh

    return 0;  
}

### Bài tập 5: Đa năng hóa toán tử

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



//cau5

//Nhữ Ngọc Minh

#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;

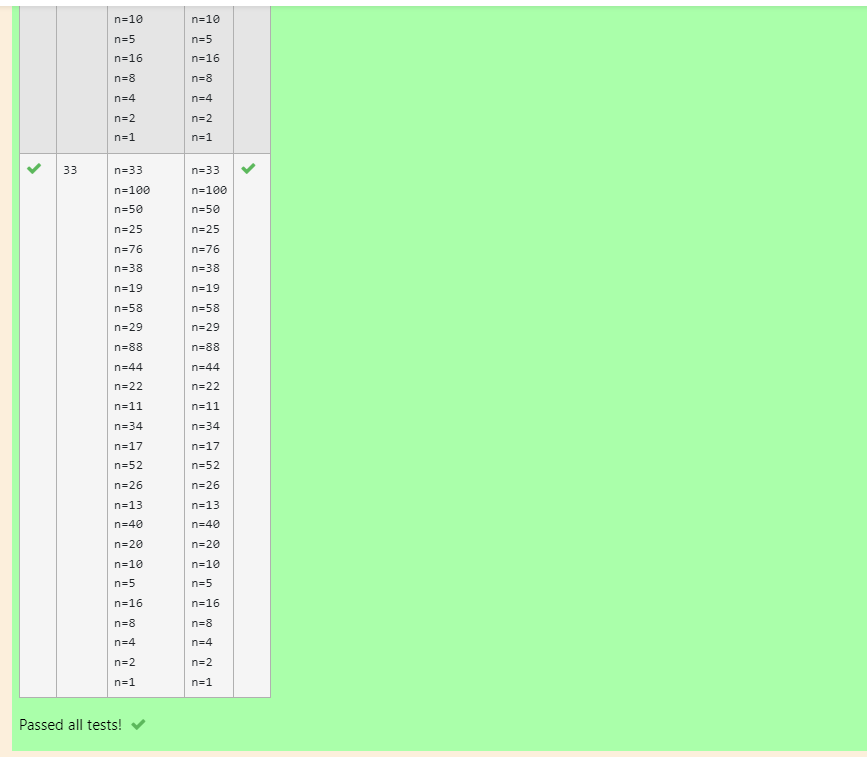
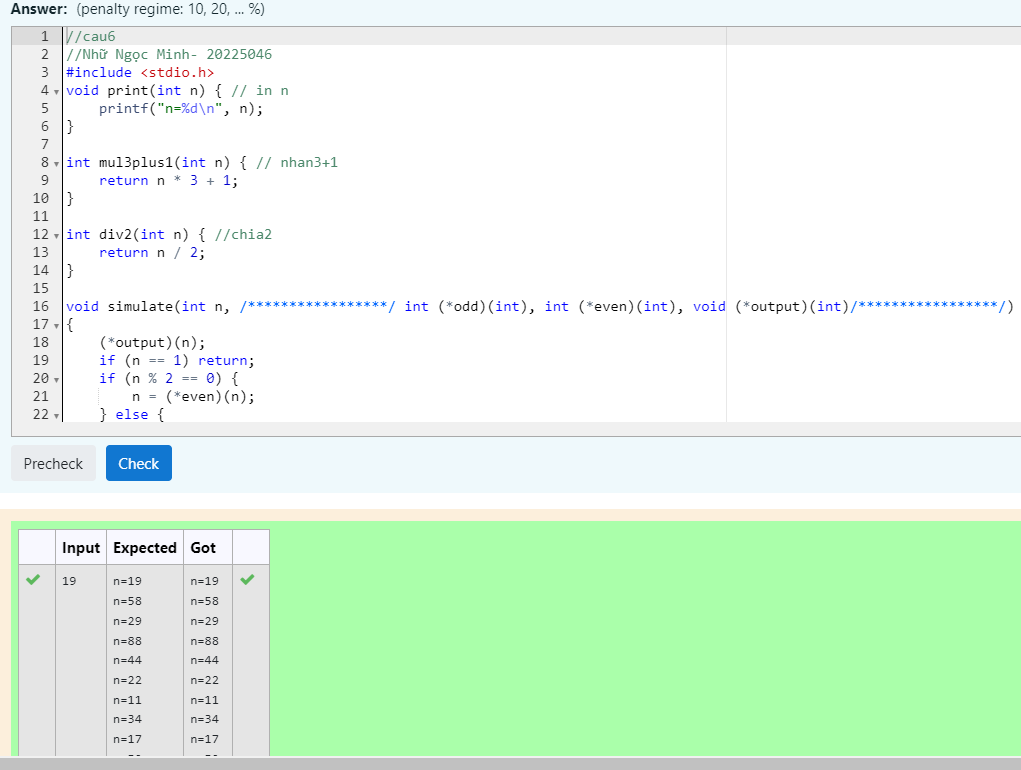
}

## 1.3 Con trỏ hàm và tham số hóa hàm

### Bài tập 6: Con trỏ hàm

Giả thuyết Collatz: bắt đầu từ số dương nn bất kỳ, nếu nn 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.



//cau6

//Nhữ Ngọc Minh- 20225046

#include <stdio.h>

void print(int n) { // in n

printf("n=%d\n", n);

}

int mul3plus1(int n) { // nhan3+1

return n \* 3 + 1;

}

int div2(int n) { //chia2

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; //khaibao

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

odd = mul3plus1;

even = div2;

int n;

scanf("%d", &n); //doc n

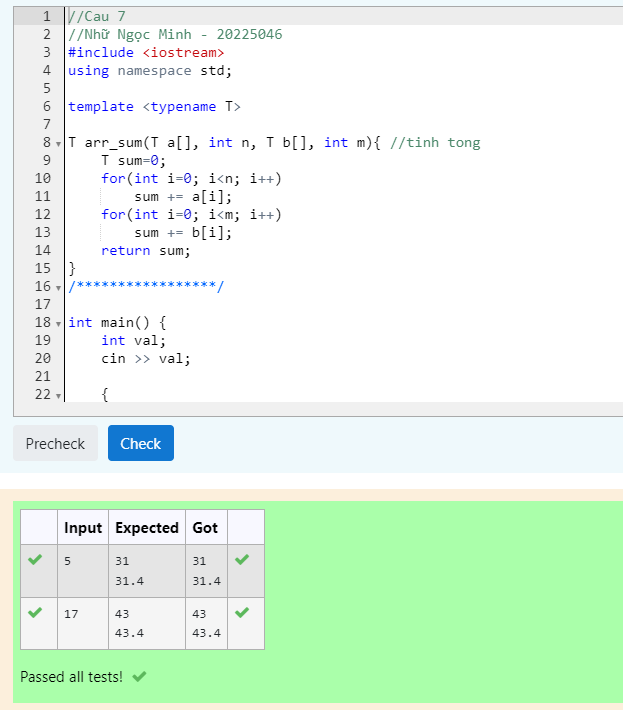
simulate(n, odd, even, print); // goi ham chung minh

return 0;

}

### Bài tập 7: Khái quát hóa hàm

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.



//Cau 7

//Nhữ Ngọc Minh - 20225046

#include <iostream>

using namespace std;

template <typename T>

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

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() {

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;

}

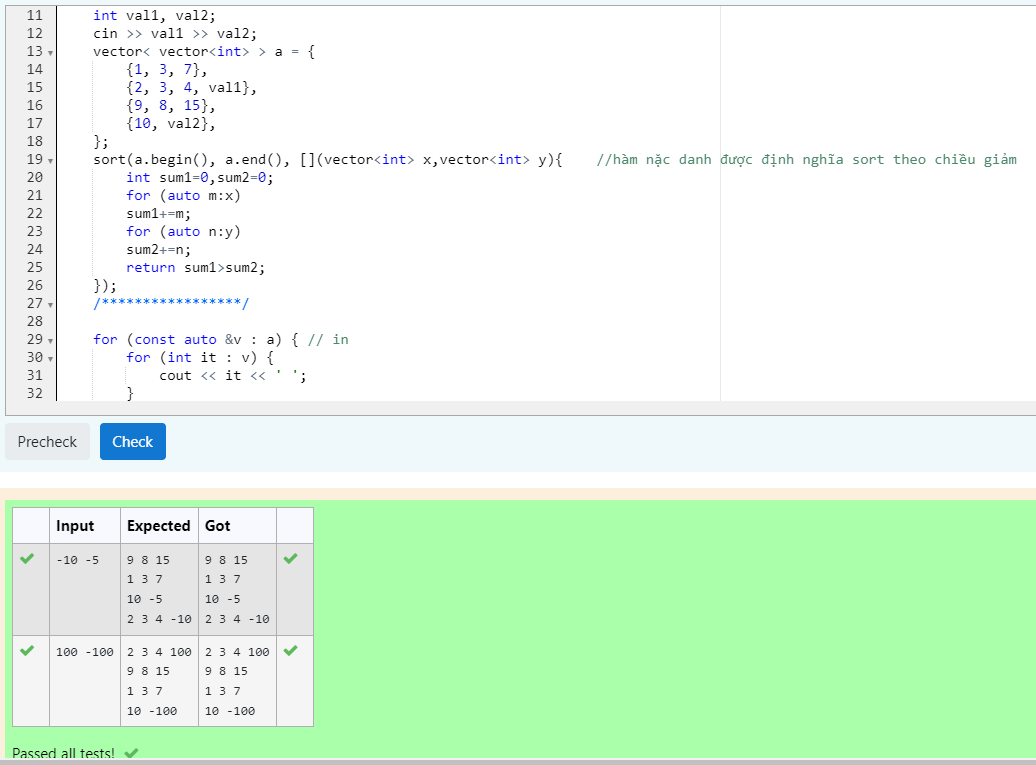
return 0;

}

# 1.4 Biểu thức lamda và hàm nặc danh

### Bài tập 8: Sắp xếp

Viết hàm so sánh cho thuật toán sắp xếp.

// Cau 8

//Nhữ Ngọc Minh- 20225046

#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

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> x,vector<int> y){ //hàm nặc danh được định nghĩa sort theo chiều giảm

int sum1=0,sum2=0;

for (auto m:x)

sum1+=m;

for (auto n:y)

sum2+=n;

return sum1>sum2;

});

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (const auto &v : a) { // in

for (int it : v) {

cout << it << ' ';

}

cout << endl;

}

return 0;

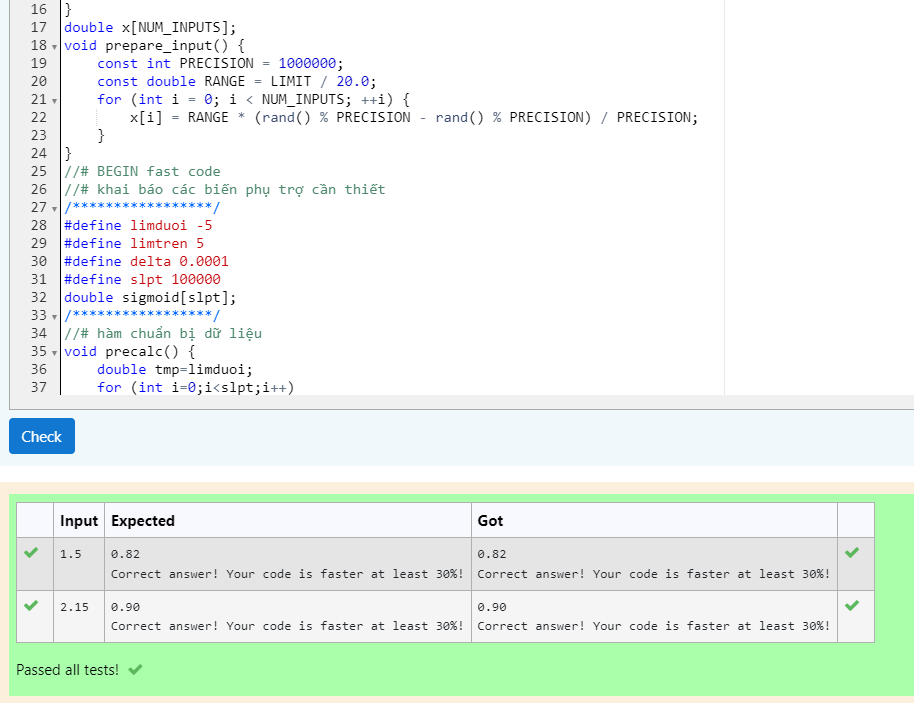
}

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

Hãy giải các bài toán sau đây một cách tối ưu nhất có thể, cố gắng sử dụng các kỹ thuật đã được học như inline, static, ...

### Bài tập 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  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.

//Cau 9

// Nhữ Ngọc Minh- 20225046

#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 biến phụ trợ cần thiết

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define limduoi -5

#define limtren 5

#define delta 0.0001

#define slpt 100000

double sigmoid[slpt];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//# hàm chuẩn bị dữ liệu

void precalc() {

double tmp=limduoi;

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

{

sigmoid[i]=1.0/(1.0+exp(-tmp)); tmp+=delta;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

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

inline double sigmoid\_fast(double x) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (x>limtren) return 0;

else if (x<limduoi) return 1;

else

{ int i=floor((x-limduoi)/delta);

return sigmoid[i]+(sigmoid[i+1]-sigmoid[i])\*(x-limduoi-delta\*i)/delta;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

//# 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() {

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 {

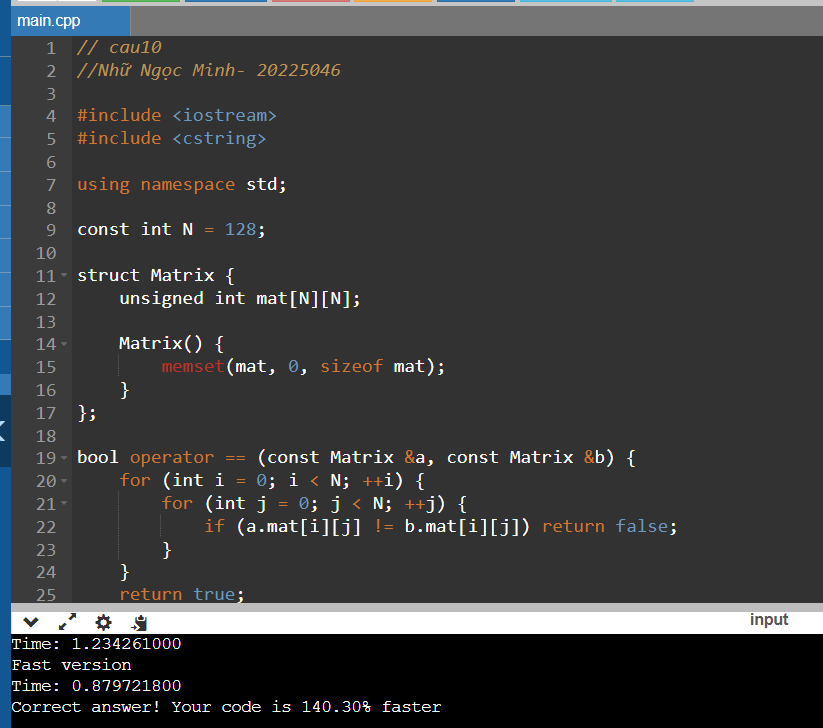
//printf("Wrong answer or your code is not fast enough!\n"); }

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 NxN 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().



// cau10

//Nhữ Ngọc Minh- 20225046

#include <iostream>

#include <cstring>

using namespace std;

const int N = 128;

struct Matrix {

unsigned int mat[N][N];

Matrix() {

memset(mat, 0, sizeof mat);

}

};

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) {

int sum = 0;

for (int k = 0; k < N; ++k) {

sum += a.mat[i][k] \* b.mat[k][j];

}

c.mat[i][j] = sum;

}

}

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;

}

# 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 N và M. Hãy tính ma trận tích C(x) = A(x) \* B(x) có bậc N+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:**

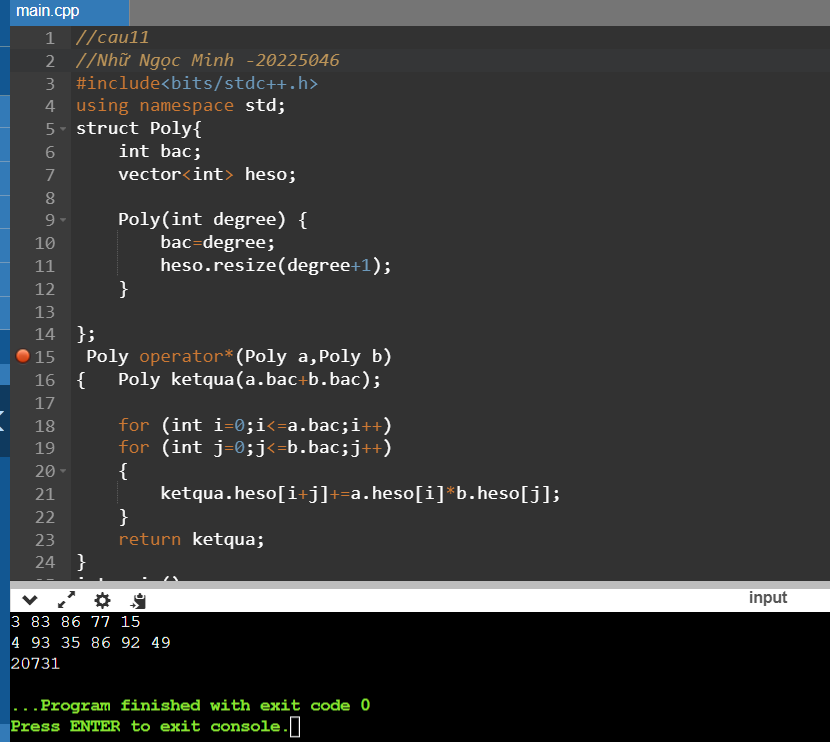
**Input:**  
3 83 86 77 15  
4 93 35 86 92 49

**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 .



//cau11

//Nhữ Ngọc Minh -20225046

#include<bits/stdc++.h>

using namespace std;

struct Poly{

int bac;

vector<int> heso;

Poly(int degree) {

bac=degree;

heso.resize(degree+1);

}

};

Poly operator\*(Poly a,Poly b)

{ Poly ketqua(a.bac+b.bac);

for (int i=0;i<=a.bac;i++)

for (int j=0;j<=b.bac;j++)

{

ketqua.heso[i+j]+=a.heso[i]\*b.heso[j];

}

return ketqua;

}

int main()

{

int n,m,tmp;

cin>>n; Poly a(n);

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

{

cin>>tmp; a.heso[i]=tmp;

}

cin>>m; Poly b(m);

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

{

cin>>tmp; b.heso[i]=tmp;

}

Poly mul(m+n);

mul=a\*b;

int tmp2=mul.heso[0];

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

tmp2=(tmp2^mul.heso[i]);

cout<<tmp2;

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 <key, value> 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.

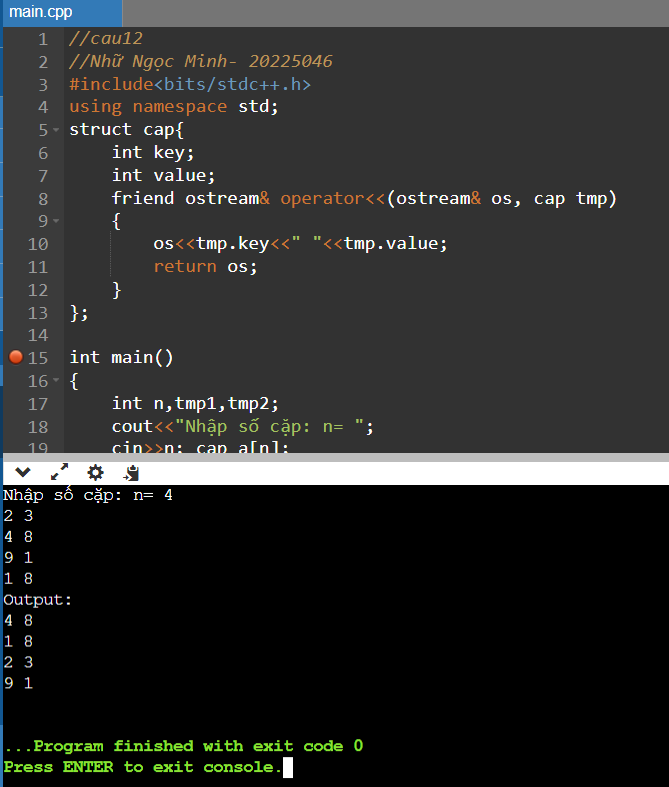
**Input:** Danh sách đầu vào. Mỗi dòng ghi một cặp giá trị key, value cách nhau bởi dấu cách (|key| ≤, |value| ≤).

**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.

**Ví dụ:**

**Input:**  
2 3  
4 8  
9 1  
1 8

**Output:**  
4 8  
1 8  
2 3  
9 1



//cau12

//Nhữ Ngọc Minh- 20225046

#include<bits/stdc++.h>

using namespace std;

struct cap{

int key;

int value;

friend ostream& operator<<(ostream& os, cap tmp)

{

os<<tmp.key<<" "<<tmp.value;

return os;

}

};

int main()

{

int n,tmp1,tmp2;

cout<<"Nhập số cặp: n= ";

cin>>n; cap a[n];

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

{

cin>>tmp1>>tmp2;

a[i]=cap(tmp1,tmp2);

}

sort(a,a+n,[](cap v1,cap v2){

if (v1.value!=v2.value) return v1.value>v2.value;

else return v1.key>v2.key;

}

);

cout<<"Output: \n";

for (auto i: a) cout<<i<<"\n";

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 . 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.

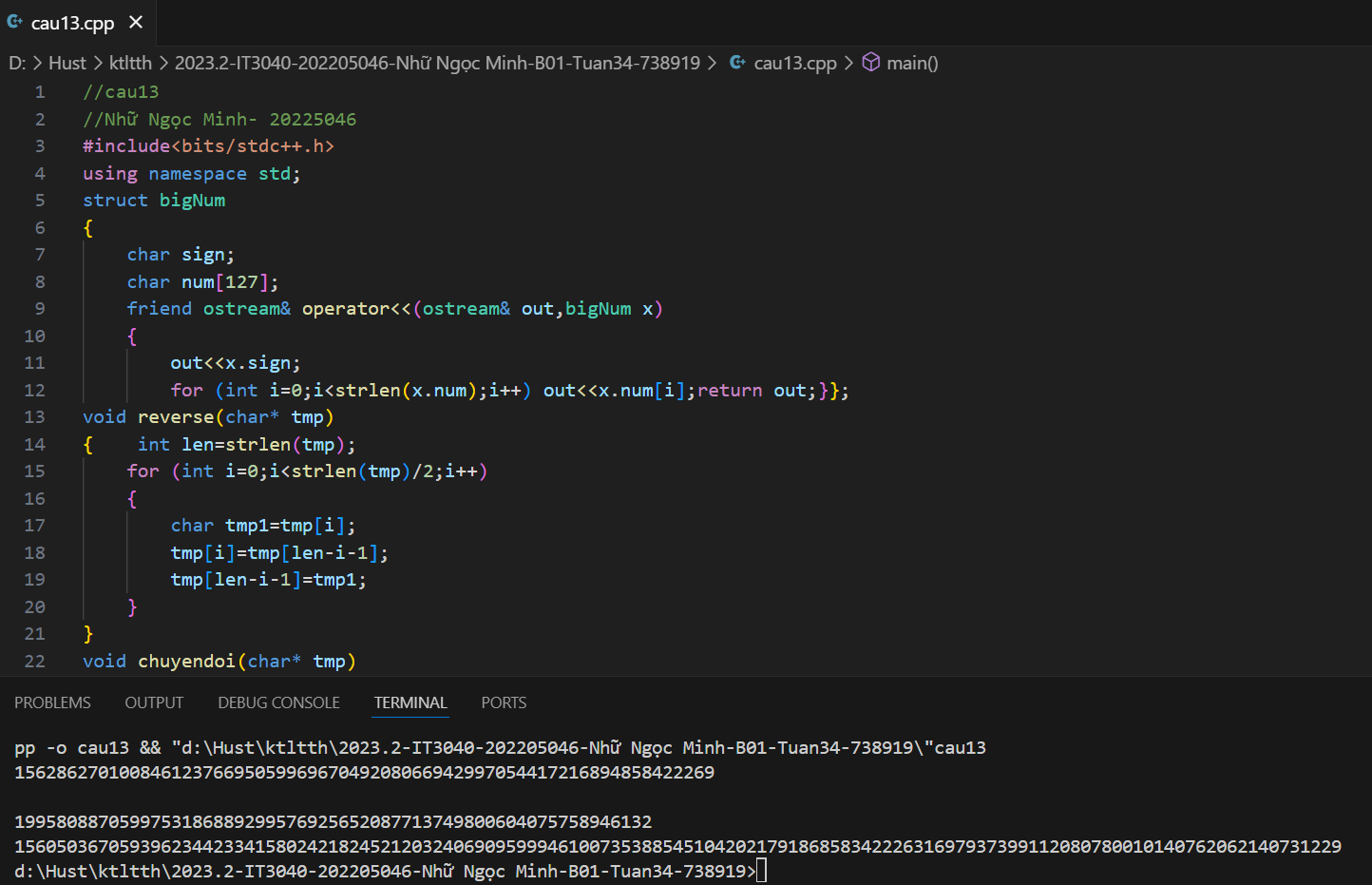
**Output:** In ra giá trị của biểu thức ab−3a+4b. Kết quả in ra một số nguyên lớn dưới dạng chuỗi ký tự có định dạng như mô tả trong dữ liệu vào.

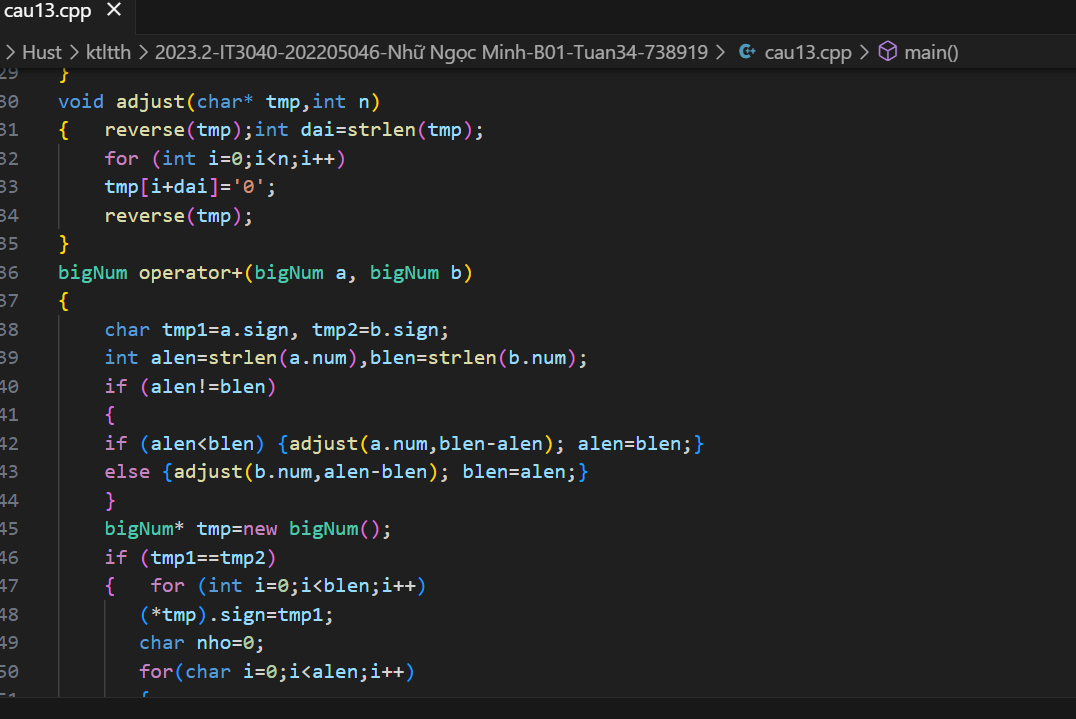
**Ví dụ:**

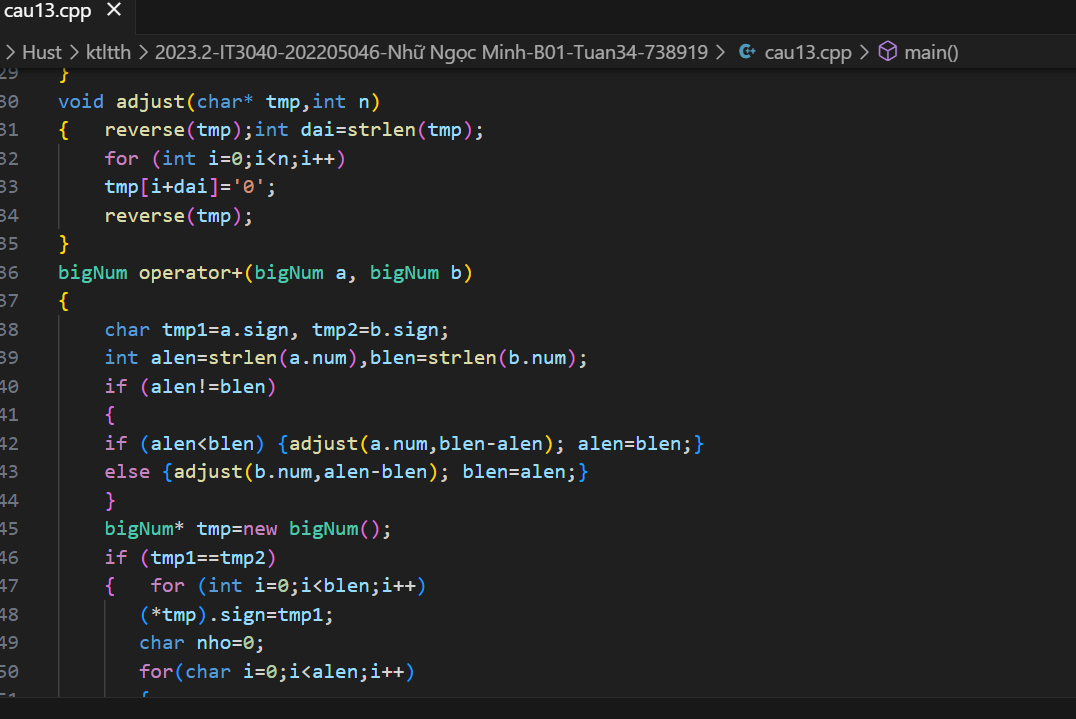
**Input:**  
0121807015  
1347227347

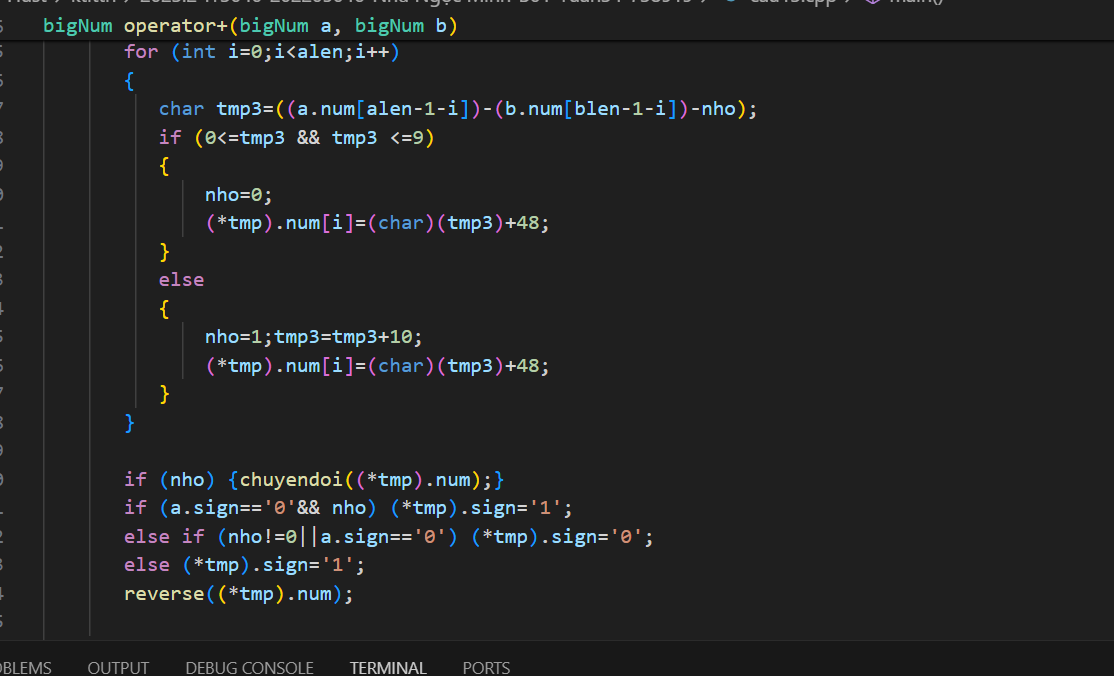
**Output:**

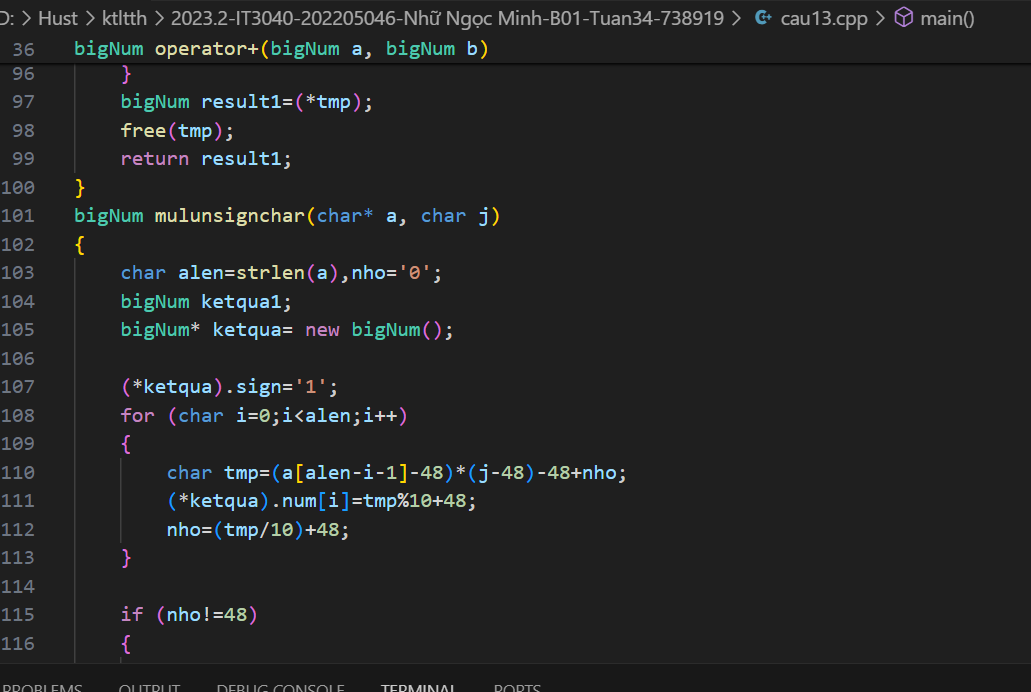
042294724910108772

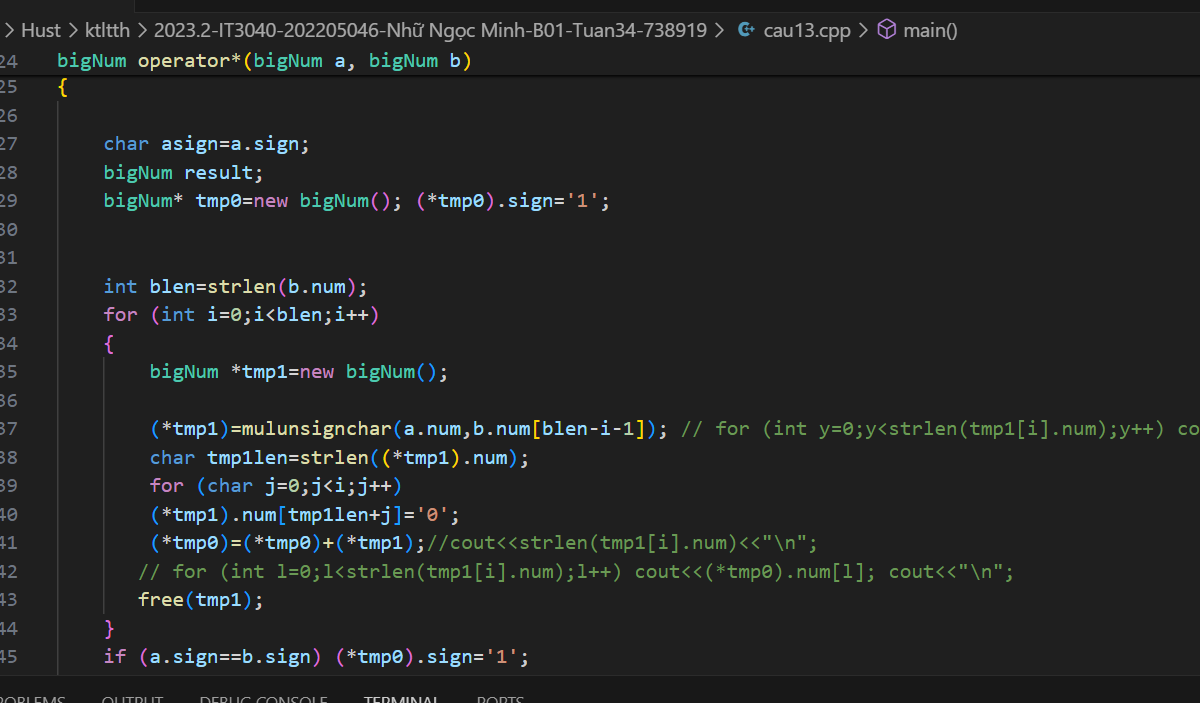


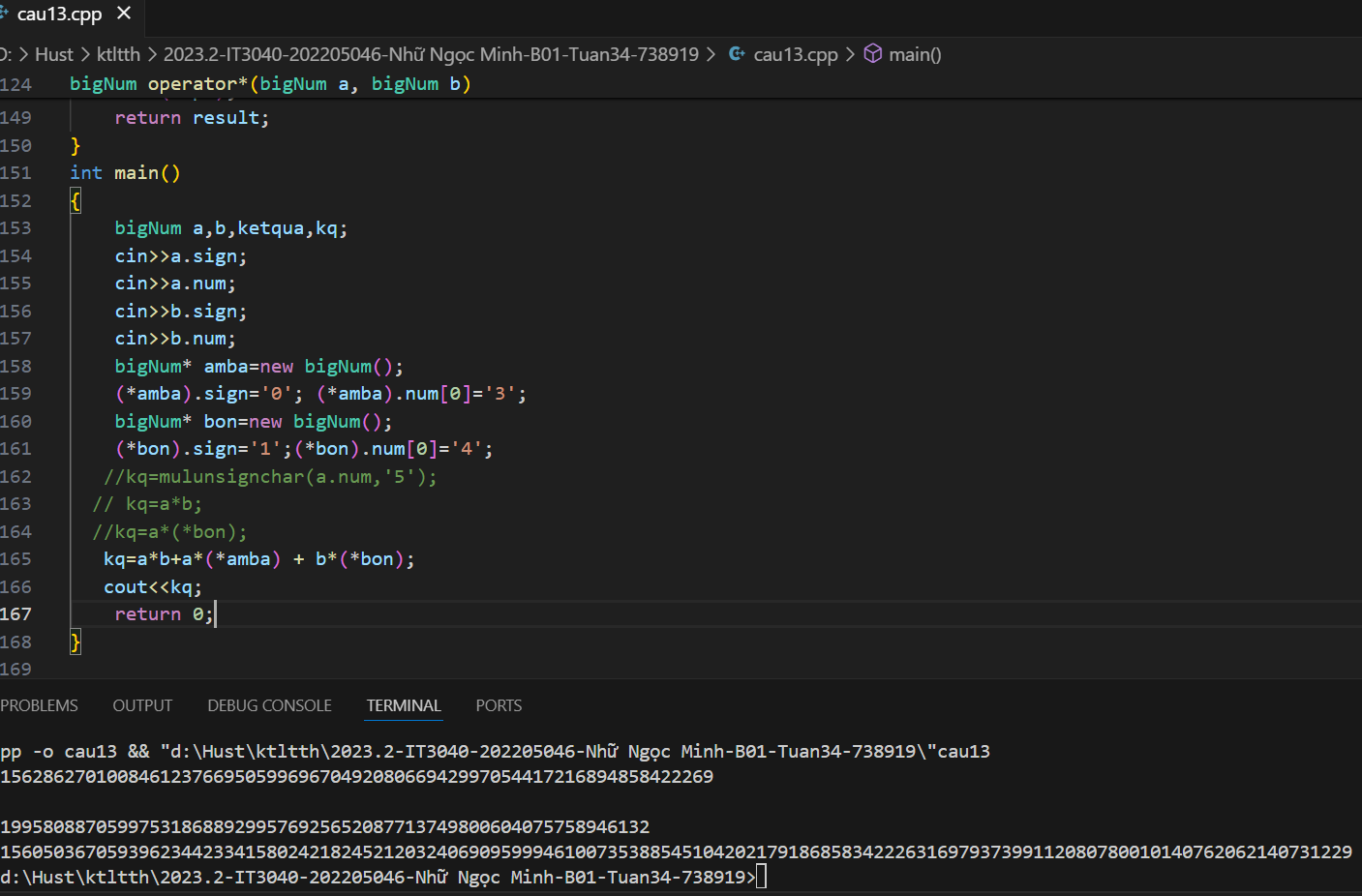












//cau13

//Nhữ Ngọc Minh- 20225046

#include<bits/stdc++.h>

using namespace std;

struct bigNum

{

    char sign;

    char num[127];

    friend ostream& operator<<(ostream& out,bigNum x)

    {

        out<<x.sign;

        for (int i=0;i<strlen(x.num);i++) out<<x.num[i];return out;}};

void reverse(char\* tmp)

{    int len=strlen(tmp);

    for (int i=0;i<strlen(tmp)/2;i++)

    {

        char tmp1=tmp[i];

        tmp[i]=tmp[len-i-1];

        tmp[len-i-1]=tmp1;

    }

}

void chuyendoi(char\* tmp)

{   int m=strlen(tmp);

    tmp[0]=106-tmp[0];

    for (int i=1;i<strlen(tmp);i++)

    {

        tmp[i]=105-tmp[i];

    }

}

void adjust(char\* tmp,int n)

{   reverse(tmp);int dai=strlen(tmp);

    for (int i=0;i<n;i++)

    tmp[i+dai]='0';

    reverse(tmp);

}

bigNum operator+(bigNum a, bigNum b)

{

    char tmp1=a.sign, tmp2=b.sign;

    int alen=strlen(a.num),blen=strlen(b.num);

    if (alen!=blen)

    {

    if (alen<blen) {adjust(a.num,blen-alen); alen=blen;}

    else {adjust(b.num,alen-blen); blen=alen;}

    }

    bigNum\* tmp=new bigNum();

    if (tmp1==tmp2)

    {   for (int i=0;i<blen;i++)

       (\*tmp).sign=tmp1;

       char nho=0;

       for(char i=0;i<alen;i++)

       {

          int tmp3=((int)(a.num[alen-1-i])+(int)(b.num[blen-1-i])-48-48+nho);

          if (0<=tmp3 && tmp3 <=9)

          {

              nho=0;

              (\*tmp).num[i]=(tmp3)+48;

          }

          else

          {

              nho=1;tmp3=tmp3-10;

              (\*tmp).num[i]=(tmp3)+48;

          }

       }

       if (nho) (\*tmp).num[alen]='1';

       reverse((\*tmp).num);

    }

    else

    {

        if (alen!=blen)

    {

    if (alen<blen) {adjust(a.num,blen-alen); alen=blen;}

    else {adjust(b.num,alen-blen); blen=alen;}

    }

       int nho=0;

       for (int i=0;i<alen;i++)

       {

          char tmp3=((a.num[alen-1-i])-(b.num[blen-1-i])-nho);

          if (0<=tmp3 && tmp3 <=9)

          {

              nho=0;

              (\*tmp).num[i]=(char)(tmp3)+48;

          }

          else

          {

              nho=1;tmp3=tmp3+10;

              (\*tmp).num[i]=(char)(tmp3)+48;

          }

       }

       if (nho) {chuyendoi((\*tmp).num);}

       if (a.sign=='0'&& nho) (\*tmp).sign='1';

       else if (nho!=0||a.sign=='0') (\*tmp).sign='0';

       else (\*tmp).sign='1';

       reverse((\*tmp).num);

    }

    bigNum result1=(\*tmp);

    free(tmp);

    return result1;

}

bigNum mulunsignchar(char\* a, char j)

{

    char alen=strlen(a),nho='0';

    bigNum ketqua1;

    bigNum\* ketqua= new bigNum();

    (\*ketqua).sign='1';

    for (char i=0;i<alen;i++)

    {

        char tmp=(a[alen-i-1]-48)\*(j-48)-48+nho;

        (\*ketqua).num[i]=tmp%10+48;

        nho=(tmp/10)+48;

    }

    if (nho!=48)

    {

        (\*ketqua).num[alen]=nho;

    }

    reverse((\*ketqua).num);

    ketqua1=\*ketqua;

    free(ketqua);

    return ketqua1;

}

bigNum operator\*(bigNum a, bigNum b)

{

    char asign=a.sign;

    bigNum result;

    bigNum\* tmp0=new bigNum(); (\*tmp0).sign='1';

    int blen=strlen(b.num);

    for (int i=0;i<blen;i++)

    {

        bigNum \*tmp1=new bigNum();

        (\*tmp1)=mulunsignchar(a.num,b.num[blen-i-1]);

        char tmp1len=strlen((\*tmp1).num);

        for (char j=0;j<i;j++)

        (\*tmp1).num[tmp1len+j]='0';

        (\*tmp0)=(\*tmp0)+(\*tmp1);

       free(tmp1);

    }

    if (a.sign==b.sign) (\*tmp0).sign='1';

    else (\*tmp0).sign='0';

    result=\*tmp0;

    free(tmp0);

    return result;

}

int main()

{

    bigNum a,b,ketqua,kq;

    cin>>a.sign;

    cin>>a.num;

    cin>>b.sign;

    cin>>b.num;

    bigNum\* amba=new bigNum();

    (\*amba).sign='0'; (\*amba).num[0]='3';

    bigNum\* bon=new bigNum();

    (\*bon).sign='1';(\*bon).num[0]='4';

   kq=a\*b+a\*(\*amba) + b\*(\*bon);

   cout<<kq;

    return 0;

}