Mục lục nội dung

[Bài thực hành số 02 – Tuần 36 3](#_Toc166929254)

[BÀI TẬP TRÊN LAP 3](#_Toc166929255)

[**Bài 2.1.**  Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông. 3](#_Toc166929256)

[**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. 4](#_Toc166929257)

[**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. 5](#_Toc166929258)

[**Bài 2.4**. Viết các hàm tính lập phương của số nguyên và số thực. 7](#_Toc166929259)

[**Bài 2.5**. Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức 9](#_Toc166929260)

[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. 13](#_Toc166929261)

[**Bài 2.7.**Viết hàm tính tổng các phần tử trong hai mảng. 15](#_Toc166929262)

[**Bài 2.8**. Viết hàm so sánh cho thuật toán sắp xếp. 17](#_Toc166929263)

[**Bài 2.9**. Tính hàm sigmoid 19](#_Toc166929264)

[**BÀI TẬP VỀ NHÀ** 26](#_Toc166929265)

[**Bài tập 2.11:** 26](#_Toc166929266)

[Cho 2 đa thức A(x) và B(x) tương ứng có bậc N và M. Hãy tính đa thức tích C(x) = A(x) \* B(x) có bậc N+M. 26](#_Toc166929267)

[**Bài tập 2.12:** 35](#_Toc166929268)

[**Bài tập 2.13:** 35](#_Toc166929269)

Mục lục hình ảnh

[Hình 1 Bài 2.1 Tính độ dài cạnh huyền 3](#_Toc166673452)

[Hình 2 Bài 2.2 Hoán vị vòng tròn 4](#_Toc166673453)

[Hình 3 Bài 2.3 Tính giá trị đa thức bằng tham số ngầm định 6](#_Toc166673454)

[Hình 4 Bài 2.4 Đa năng hóa hàm tính lập phương 7](#_Toc166673455)

[Hình 5 Bài 2.5 Đa năng hóa toán tử với số phức 10](#_Toc166673456)

[Hình 6 Bài 2.6 Con trỏ hàm kiểm định giả thuyết 13](#_Toc166673457)

[Hình 7 Bài 2.7 Template Function tính tổng các phần tử mảng 16](#_Toc166673458)

[Hình 8 Bài 2.8 Biểu thức Lamda sắp xếp vecto 17](#_Toc166673459)

[Hình 9. Bài 2.9. Tính sigmoid 22](#_Toc166673460)

# Bài thực hành số 02 – Tuần 36

# BÀI TẬP TRÊN LAP

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

Ảnh có chứa văn bản, Phông chữ, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 1 Bài 2.1 Tính độ dài cạnh huyền

#include <stdio.h>

#include <math.h>

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

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

return sqrt(x\*x + y\*y); //ch^2 = x^2 + y^2;

}

int main(){

float x, y; //khai báo x,y

scanf("%f%f", &x, &y); //nhập x,y

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

printf("z = %.2f\n", z); //in cạnh huyền

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.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màu trắng

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 2 Bài 2.2 Hoán vị vòng tròn

#include <stdio.h>

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

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

int temp = x; //lưu giá trị cũ của x

x=y; //gán y cho x

y=z; //gán z cho y

z=temp; //gán giá trị cũ của x cho z

}

int main() {

int x, y, z; //khai báo biến

//# Nhập 3 số nguyên

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

printf("Before: %d, %d, %d\n", x, y, z); //in giá trị cũ

rotate(x, y, z); //gọi hàm

printf("After: %d, %d, %d\n", x, y, z); //in giá trị sau thay đổi

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.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, Phông chữ, số, ảnh chụp màn hình

Mô tả được tạo tự động

#include <stdio.h>

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

//# Viết hàm get\_value

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

{

return a\*x\*x + b\*x + c; //trả về giá trị ax^2 + bx + c

}

int main(){

int x; //khai báo x

scanf("%d", &x); //nhập vào 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

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

printf("a=2, b=1, c=0: %d\n", get\_value(x)); //gọi hàm get\_value với a,b,c mặc định

printf("a=%d, b=1, c=0: %d\n", a, get\_value(x, a)); //gọi hàm get\_value với b,c mặc định, a được nhập từ bàn phím

printf("a=%d, b=%d, c=0: %d\n", a, b, get\_value(x, a, b)); //gọi hàm get\_value với c mặc định, a,b được nhập từ bàn phím

printf("a=%d, b=%d, c=%d: %d\n", a, b, c, get\_value(x, a, b, c)); //gọi hàm get\_value với a,b,c được nhập từ bàn phím

return 0;

}

Hình 3 Bài 2.3 Tính giá trị đa thức bằng tham số ngầm định

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

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màu trắng

Mô tả được tạo tự độngẢnh có chứa văn bản, Phông chữ, hàng, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 4 Bài 2.4 Đa năng hóa hàm tính lập phương

#include <stdio.h>

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

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; //khai báo n

double f; //khai báo f

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

printf("Int: %d\n", cube(n)); //in giá trị n^3

printf("Double: %.2lf\n", cube(f)); //in giá trị f^3

return 0;

}

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

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, Phông chữ, số, hàng

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, số, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 5 Bài 2.5 Đa năng hóa toán tử với số phức

#include <iostream>

#include <ostream>

#include <math.h>

#include <iomanip>

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

using namespace std;

struct Complex { //định nghĩa số phức

double real;

double imag;

};

Complex operator + (Complex a, Complex b) { //đa năng hóa phép cộng

Complex c; //c=a+b

c.real = a.real + b.real; //c.thực = a.thực + b.thực

c.imag = a.imag + b.imag; //c.ảo = a.ảo + b.ảo

return c; //trả về c;

}

Complex operator - (Complex a, Complex b) { //đa năng hóa phép trừ

Complex c; //c=a-b

c.real = a.real - b.real; //c.thực = a.thực - b.thực

c.imag = a.imag - b.imag; //c.ảo = a.ảo - b.ảo

return c; //trả về c

}

Complex operator \* (Complex a, Complex b) { //đa năng hóa phép nhân

Complex c; //c=a\*b

c.real = a.real \* b.real - a.imag \* b.imag; //c.thực = a.thực \* b.thực - a.ảo \* b.ảo

c.imag = a.real \* b.imag + b.real \* a.imag; //c.ảo = a.thực \* b.ảo + b.thực \* a.ảo

return c; //trả về c

}

Complex operator / (Complex a, Complex b) { //đa năng hóa phép chia

Complex c; //c=a/b

c.real = (a.real \* b.real + a.imag \* b.imag)/(b.real \* b.real + b.imag \* b.imag); //c.thực = (a.thực\*b.thực+a.ảo\*b.ảo)/((mô đun b)^2)

c.imag = (a.imag \* b.real - a.real \* b.imag)/(b.real \* b.real + b.imag \* b.imag); //c.ảo = (a.ảo\*b.thực-a.thực\*b.ảo)/((mô đun b)^2)

return c;

}

ostream& operator << (ostream& out, const Complex &a) { //đa năng hóa toán tử >>

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; //khai báo phần thực, phần ảo

cin >> real\_a >> img\_a; //nhập vào phần thực, ảo của số phức a

cin >> real\_b >> img\_b; //nhập vào phần thực, ảo của số phức b

Complex a{real\_a, img\_a}; //khai báo số phức a

Complex b{real\_b, img\_b}; //khai báo số phức b

cout << a << " + " << b << " = " << a + b << endl; //in a+b

cout << a << " - " << b << " = " << a - b << endl; //in a-b

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

cout << a << " / " << b << " = " << a / b << endl; //in a/b

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.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, số, màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, số

Mô tả được tạo tự động

Hình 6 Bài 2.6 Con trỏ hàm kiểm định giả thuyết

#include <stdio.h>

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

void print(int n) { //in

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

}

int mul3plus1(int n) { //nhân 3 cộng 1

return n \* 3 + 1;

}

int div2(int n) { //chia 2

return n / 2;

}

// khai báo các tham số cho các con trỏ hàm odd, even và output

void simulate(int n,int (\*odd)(int),int (\*even)(int),void (\*output)(int)) //truyền vào hàm số nguyên n,

//con trỏ hàm odd nhận tham số đầu vào là int, trả về int

//con trỏ hàm even nhận tham số đầu vào là int, trả về int

{ //con trỏ hàm output nhận tham số đầu vào là int, trả về void

(\*output)(n); //thực hiện hàm được trỏ bởi con trỏ output

if (n == 1) return; //nếu n = 1, kết thúc

if (n % 2 == 0) { //nếu n chẵn, thực hiện hàm được trỏ bởi con trỏ even

n = (\*even)(n);

} else { //nếu n lẻ, thực hiện hàm được trỏ bởi con trỏ odd

n = (\*odd)(n);

}

simulate(n, odd, even, output); //tiếp tục đệ quy cho đến khi n = 1

}

int main() {

int (\*odd)(int) = NULL; //khai báo con trỏ hàm odd trỏ vào null

int (\*even)(int) = NULL; //khai báo con trỏ hàm even trỏ vào null

odd = mul3plus1; //odd trỏ vào hàm mul3plus1

even = div2; //even trỏ vào hàm div2

int n; //khai báo n

scanf("%d", &n); //nhập vào n

simulate(n, odd, even, print); //mô phỏng quá trình của n

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.**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biểu đồ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, hàng

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, hàng, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 7 Bài 2.7 Template Function tính tổng các phần tử mảng

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

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, hàng, số

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động

Hình 8 Bài 2.8 Biểu thức Lamda sắp xếp vecto

#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

using namespace std;

int main() {

int val1, val2; //khai báo val1, val2

cin >> val1 >> val2; //nhập vào val1, val2

vector< vector<int> > a = { //khai báo vector các vector a

{1, 3, 7},

{2, 3, 4, val1},

{9, 8, 15},

{10, val2},

};

//# sắp xếp các vector trong a theo tổng các phần tử giảm dần

sort(a.begin(),a.end(), //gọi hàm sort

[](vector<int> &p, vector<int> &q) //tham số thứ 3 truyền vào hàm sort

{ //là hàm để định nghĩa luật sắp xếp

return accumulate(p.begin(),p.end(),0)

> accumulate(q.begin(),q.end(),0);

});

//luật: tổng phần tử vector trước lớn hơn tổng phần tử vector sau

//sử dụng hàm accumulate để tính tổng

for (const auto &v : a) { //với mỗi vector trong a

for (int it : v) { //với mỗi phần tử trong vector

cout << it << ' '; //in ra

}

cout << endl;

}

return 0;

}

## **Bài 2.9**. Tính hàm sigmoid

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, số

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 9. Bài 2.9. Tính sigmoid

#include <vector>

#include <algorithm>

#include <cmath>

#include <ctime>

#include <algorithm>

#include <cstdio>

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

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

const int N = 1000000;

double y[N+1];

//# hàm chuẩn bị dữ liệu

void precalc() {

double delta = (LIMIT \* 2.0)/N;

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

{

y[i] = sigmoid\_slow(-LIMIT + delta\*i);

}

}

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

inline double sigmoid\_fast(double x) {

if (x < -LIMIT) return 0;

if (x >= LIMIT) return 1;

static double delta = (LIMIT \* 2.0)/N;

int i = (x + LIMIT) / delta;

return (x-(-LIMIT + (delta \* i))) / delta \* (y[i+1] - y[i]) + y[i];

}

//# 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 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 VỀ NHÀ**

Chụp ảnh kết quả của tất cả các test.

## **Bài tập 2.11:**

Cho 2 đa thức A(x) và B(x) tương ứng có bậc N và M. Hãy tính đa thức tích C(x) = A(x) \* B(x) có bậc N+M.

**Dữ liệu vào:**

Gồm 2 dòng biểu diễn các đa thức A(x) và B(x), mỗi dòng bao gồm

* Số đầu tiên N là một số nguyên dương tương ứng với bậc của đa thức;
* N+1 số nguyên tiếp theo là các hệ số của đa thức, số thứ i là hệ số của xi-1.

**Kết quả:**Ghi ra một số nguyên duy nhất là **XOR** của các hệ số của đa thức C(x).

Ảnh có chứa văn bản, ảnh chụp màn hình, biên lai, Phông chữ

Mô tả được tạo tự động

Hình 10 Bài 2.11

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự động

#include<bits/stdc++.h>

using namespace std;

typedef complex<double> base;

typedef vector<base> vb;

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

void input(int &n, int &m, vb &x, vb &y){ //nhập hai đa thức

cin >> n;

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

int tmp;

cin >> tmp;

base mycomplex(tmp,0);

x.push\_back(mycomplex);

}

cin >> m;

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

int tmp;

cin >> tmp;

base mycomplex(tmp,0);

y.push\_back(mycomplex);

}

}

void fft(vb & a, bool revert){ //biến đổi fourier nhanh

//tham số revert: true: biểu diễn hệ số -> biểu diễn giá trị

//false: biểu diễn giá trị -> biểu diễn hệ số

int n = (int)a.size();

for(int i=1, j=0; i<n; ++i){

int bit = n >> 1;

while(j>=bit){

j =j- bit;

bit=bit>>1;

}

j =j+ bit;

if (i < j)

swap (a[i], a[j]);

}

for(int len=2; len<=n; len<<=1){

double ang = 2\*M\_PI/len;

if(revert==1) ang=-ang;

complex<double> wlen (cos(ang), sin(ang));

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

complex<double> w (1);

for (int j=0; j<len/2; ++j) {

complex<double> u = a[i+j], v = a[i+j+len/2] \* w;

a[i+j] = u + v;

a[i+j+len/2] = u - v;

w =w \* wlen;

}

}

}

if(revert==1)

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

a[i] /=n;

}

int multiPly(int n, int m, vb x, vb y){

int p = 1;

while(p < max (n, m)) p=p<<1;

p=p<<1;

x.resize(p);

y.resize(p);

fft(x, false);

fft(y, false);

vector<base> h(p);

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

h[i] =x[i]\*y[i];

fft (h, true);

int res = (int)(real(h[0])+0.5);

for (int i=1; i<=p; i++){ //xor các hệ số đa thức kết quả

res = res ^ (int)(real(h[i])+0.5);

}

return res;

}

int main(){

int n, m;

vector<base> x,y;

input(n,m,x,y);

cout << multiPly(n,m,x,y);

}

***Case 1:***

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

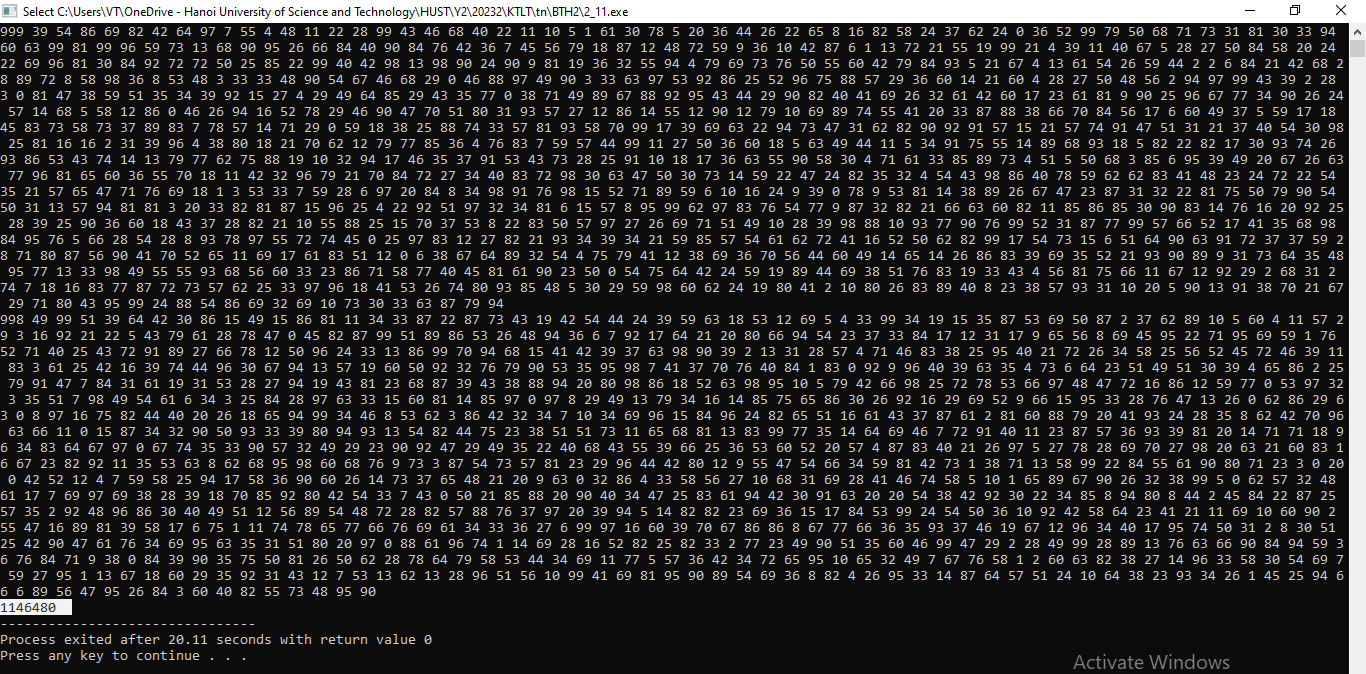
Mô tả được tạo tự động

***Case 2:***

Ảnh có chứa văn bản, Phông chữ, phần mềm, ảnh chụp màn hình

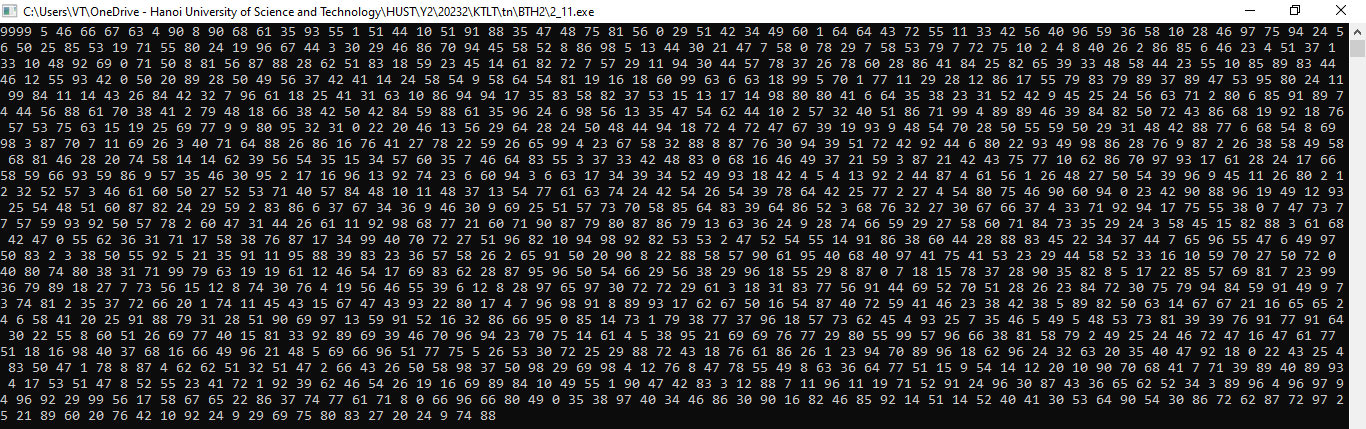
Mô tả được tạo tự động

***Case 3:***



***Case 4:***

***Input:***



Ảnh có chứa ảnh chụp màn hình, mẫu, tác phẩm nghệ thuật

Mô tả được tạo tự động

***Output:***

Ảnh có chứa ảnh chụp màn hình, văn bản, Phông chữ

Mô tả được tạo tự động

***Case 5:***

***Input:***

***Đa thức 1:***

Ảnh có chứa ảnh chụp màn hình, văn bản, mẫu, Hình chữ nhật

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, mẫu, Đối xứng, tác phẩm nghệ thuật

Mô tả được tạo tự độngẢnh có chứa mẫu, trang phục, vải vóc, tác phẩm nghệ thuật

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, mẫu, tác phẩm nghệ thuật

Mô tả được tạo tự động

***Đa thức 2:***

Ảnh có chứa mẫu, trang phục, vải vóc, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa mẫu, ảnh chụp màn hình, Đối xứng, Nhiều màu sắc

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, mẫu, Đối xứng, tác phẩm nghệ thuật

Mô tả được tạo tự độngẢnh có chứa mẫu, ảnh chụp màn hình, vải vóc, tác phẩm nghệ thuật

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, tác phẩm nghệ thuật

Mô tả được tạo tự động

***Output:***

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

## **Bài tập 2.12:**

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|  ≤109 , |value|  ≤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.

Ảnh có chứa hàng, văn bản, biểu đồ, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 11. Bài 2.12

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, Phần mềm đa phương tiện, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 12. Bài 2.12

#include <iostream>

#include <vector>

#include <utility>

#include <algorithm>

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

using namespace std;

vector<pair<int,int>> a; //khai báo vector lưu các cặp số

void input() //nhập dữ liệu

{

int key,value;

while (cin>>key && cin>>value)

{

pair<int,int> temp = make\_pair(key, value);

a.push\_back(temp);

}

}

void output() //xuất dữ liệu

{

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

{

cout<<a[i].first<<" "<<a[i].second<<endl;

}

}

int main()

{

input();

sort(a.begin(),a.end(),[] (pair<int,int>p, pair<int,int> q) //sử dụng hàm nặc danh định nghĩa sắp xếp

{

if (p.second != q.second) return p.second > q.second; //nếu value khác nhau, so sánh value

else return p.first > q.first; //ngược lại, so sánh key

});

output();

return 0;

}

**Case 1:**

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

**Case 2:**

***Input:***

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thuật in máy

Mô tả được tạo tự động

***Output:***



Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, Phông chữ, văn bản, thuật in máy

Mô tả được tạo tự động

**Case 3:**

***Input, output:***

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

**Case 4:**

***Input:***

Ảnh có chứa văn bản, ảnh chụp màn hình, sách, mẫu

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, sách, mẫu, văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, thuật in máy

Mô tả được tạo tự động

***Output:***

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thuật in máy

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thuật in máy

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

**Case 5:**

***Input:***

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, màu đen, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen, thiết kế

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, màu đen, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màu đen

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, màu đen, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, màu đen

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen, thiết kế

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, màu đen, văn bản

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen, thiết kế

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, màu đen, văn bản

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động Ảnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, màu đen, Phông chữ

Mô tả được tạo tự động

***Output:***

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, Hình chữ nhật

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen, thiết kế

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, màu đen, văn bản

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, màu đen

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, Hình chữ nhật

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

## **Bài tập 2.13:**

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];*

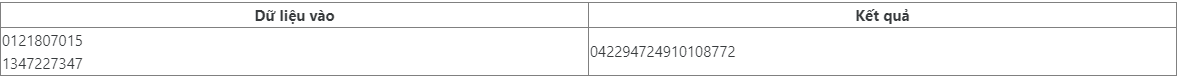
*};*

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  a  và  b , mỗi dòng chứa 1 chuỗi ký tự mô tả 1 số nguyên lớn không vượt quá  10100 . 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:**

 Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

#include<bits/stdc++.h>

using namespace std;

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

Nguyễn Kim Cường-20224939-738918-2023.2

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

struct bigNum { //định nghĩa struct bignum

char sign;

char num[1001];

};

//nhập số lớn

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[1000-lens1+i+1] = tmp[i+1];

}

for(int i=0; i<1000-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[1000-lens2+i+1] = tmp[i+1];

}

for(int i=0; i<1000-lens2+1; i++) num2.num[i] = '0';

}

//cộng hai số

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

int c = 0; //nhớ c

for(int i=1000; i>=0; i--){

int tmp = (int)num1[i] - 48 + (int)num2[i] - 48 + c;

c = tmp / 10;

res[i] = tmp % 10 + 48;

}

}

//trừ hai số (num1>num2)

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

int c = 0;

for(int i=1000; 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;

}

}

}

//nhân hai số

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

//khởi tạo kết quả

for(int i=0; i<1001; i++) res[i] = '0';

for(int i=1000; i>=0; i--){

//khởi tạo số rỗng

char tmp[1001];

int k;

for(k = 0; k < i; k++)

tmp[1000-k] = '0';

int c = 0, sum = 0; //nhân có nhớ c

for(int j=1000; 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 num1 >= num2

bool check(char \*num1, char \*num2){

int foo1, foo2;

for(foo1 = 0; foo1 < 1001; foo1++){

if(num1[foo1] != '0') break;

}

for(foo2 = 0; foo2 < 1001; 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 < 1001){

if(num1[foo] < num2[foo]) return false;

else if (num1[foo] > num2[foo]) return true;

else {

foo++;

}

}

}

return true;

}

//đa năng hóa toán tử "+"

bigNum operator + (bigNum num1, bigNum num2){

bigNum res;

if(num1.sign == '1' && num2.sign == '1'){ //+ hai số dương

res.sign = '1';

add(res.num,num1.num,num2.num);

return res;

} else if(num1.sign == '1' && num2.sign == '0'){ //+ số dương với số âm

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'){ //+ số âm với số dương

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;

}

}

//đa năng hóa toán tử "-"

bigNum operator - (bigNum num1, bigNum num2){

bigNum res;

if(num1.sign == '1' && num2.sign == '0'){ //trừ số dương với số âm

num2.sign = '1';

res = num1 + num2;

return res;

} else if(num1.sign == '1' && num2.sign == '1'){ //trừ hai số dương

num2.sign = '0';

res = num1 + num2;

return res;

} else if(num1.sign == '0' && num2.sign == '1'){ //trừ số âm với số dương

num2.sign = '0';

res = num1 + num2;

return res;

} else { //trừ hai số âm

num2.sign = '1';

res = num1 + num2;

return res;

}

}

//đa năng hóa toán tử \*

bigNum operator \* (bigNum num1, bigNum num2){

bigNum res;

if(num1.sign == '1' && num2.sign == '1'){ //nhân hai số dương

res.sign = '1';

multi(res.num,num1.num,num2.num);

return res;

} else if(num1.sign == '1' && num2.sign == '0'){ //nhân số dương và số âm

res.sign = '0';

multi(res.num,num1.num,num2.num);

return res;

} else if(num1.sign == '0' && num2.sign == '1'){ //nhân số âm và số dương

res.sign = '0';

multi(res.num,num1.num,num2.num);

return res;

} else { //nhân hai số âm

res.sign = '1';

multi(res.num,num1.num,num2.num);

return res;

}

}

//in số lớn

void printBigNumber(bigNum number){

cout << number.sign;

int start;

for(start=0; start<1001; start++)

if(number.num[start] != '0') break;

for(int i = start; i<1001; i++)

cout << number.num[i];

}

int main(){

bigNum num1, num2; //khai báo a và b

input(num1,num2);

bigNum so3, so4; //khởi tạo số 3 và 4

so3.sign = '1', so4.sign = '1';

for(int i=0; i<1000; i++){

so3.num[i] = '0';

so4.num[i] = '0';

}

so3.num[1000] = '3';

so4.num[1000] = '4';

bigNum res = num1\*num2 - so3 \* num1 + so4 \* num2; //res = a\*b - 3a +4b

printBigNumber(res); //in

}

**Case 1:**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

**Case 2:**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

**Case 3:**

Ảnh có chứa văn bản, phần mềm, Phông chữ, Website

Mô tả được tạo tự động

**Case 4:**

Ảnh có chứa văn bản, phần mềm, Website, Trang web

Mô tả được tạo tự động