

ĐẠI HỌC BÁCH KHOA HÀ NỘI  
TRƯỜNG CNTT&TT

\*\*\*\*\*  \*\*\*\*\*



**BÁO CÁO THỰC HÀNH  
HỌC PHẦN: KỸ THUẬT LẬP TRÌNH**

**Bài 1 – Tuần 9**

Sinh viên thực hiện: **Tạ Quang Phổ**  
MSSV: **20215450**  
Mã lớp: **IT3040 – 732830**

Giảng viên hướng dẫn: **ThS. Lê Thị Hoa**

\_\_\_. Năm học 2023-2024 .\_\_

## Contents

Bài thực hành số 2 – Tuần 9 .....	3
<b>Bài tập 2.1.</b> Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông. .....	3
<b>Bài tập 2.2.</b> 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. ....	5
<b>Bài tập 2.3.</b> Viết chương trình yêu cầu nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị $ax^2+bx+c$ với a, b, c định sẵn. ....	8
<b>Bài tập 2.4.</b> Viết các hàm tính lập phương của số nguyên và số thực. ....	11
<b>Bài tập 2.5.</b> Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức .....	14
<b>Bài tập 2.6.</b> 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 $n=1$ . ....	18
<b>Bài tập 2.7.</b> 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.....	22
<b>Bài tập 2.8.</b> Viết hàm so sánh cho thuật toán sắp xếp. ....	25
<b>Bài tập 2.9:</b> 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 <sup>-6</sup> và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản. ....	27
<b>Bài tập 2.11:</b> 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 $N+M-1$ . ....	33
<b>Bài tập 2.12:</b> 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. ....	38
<b>Bài tập 2.13:</b> 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: .....	48

## Bài thực hành số 2 – Tuần 9

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

```

1 #include <stdio.h>
2 #include <math.h>
3
4 float get_hypotenuse(float x, float y) {
5     //*****//
6     /* Ta Quang Pho - 20215450 */
7
8     return (sqrt(x*x + y*y)); // Theo định lý Pytago
9
10    //*****//
11 }
12
13 int main(){
14     float x, y;
15     scanf("%f%f", &x, &y);
16
17     float z = get_hypotenuse(x, y); // Gán giá trị của cạnh huyền
18                                     // ứng với 2 cạnh góc vuông là x và y
19                                     // cho biến z
20     printf("z = %.2f\n", z);
21
22     return 0;
23 }

```

	Input	Expected	Got	
✓	3 4	z = 5.00	z = 5.00	✓
✓	5 6	z = 7.81	z = 7.81	✓

Passed all tests! ✓

Mã nguồn:

```
#include <stdio.h>
```

```
int main(){
```

```
    int x, y, z;
```

```
    int* ptr;
```

Tạ Quang Phổ - 20215450

```
printf("Enter three integers: ");

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

printf("\nThe three integers are:\n");

ptr = &x;

printf("x = %d\n", *ptr);


//*****//

/* Ta Quang Pho - 20215450 */

ptr = &y;           //gán địa chỉ của biến y cho con trỏ ptr

printf("y = %d\n", *ptr);    //in ra giá trị ô nhớ được trỏ bởi con trỏ ptr

ptr = &z;           //gán địa chỉ của biến z cho con trỏ ptr

printf("z = %d\n", *ptr);    //in ra giá trị ô nhớ được trỏ bởi con trỏ ptr

//*****//

return 0;

}
```

**Bài tập 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.

```

1  #include <iostream>
2
3  using namespace std;
4
5  void rotate(int &x, int &y, int &z) {    // Truyền tham chiếu
6      //*****//
7      /* Ta Quang Pho - 20215450 */
8
9      int tmp = x;                      // Tạo biến trung gian để lưu giá trị x
10     x = y;                            // Gán giá trị x = giá trị y
11     y = z;                            // Gán giá trị y = giá trị z
12     z = tmp;                          // Gán giá trị z = giá trị trung gian
13     // (= giá trị ban đầu của x)
14
15     //*****//
16 }
17
18 int main() {
19     int x, y, z;
20
21     //# Nhập 3 số nguyên
22     //*****//
23     /* Ta Quang Pho - 20215450 */
24
25     scanf("%d%d%d", &x, &y, &z);
26
27     //*****//
28
29     printf("Before: %d, %d, %d\n", x, y, z);
30
31     rotate(x, y, z);                  // Gọi hàm rotate
32
33     printf("After: %d, %d, %d\n", x, y, z);
34
35     return 0;
36 }

```

	Input	Expected	Got	
✓	3 4 5	Before: 3, 4, 5 After: 4, 5, 3	Before: 3, 4, 5 After: 4, 5, 3	✓
✓	5 7 9	Before: 5, 7, 9 After: 7, 9, 5	Before: 5, 7, 9 After: 7, 9, 5	✓

Passed all tests! ✓

Mã nguồn:

```
#include <iostream>
```

```
using namespace std;
```

```
void rotate(int &x, int &y, int &z) { // Truyền tham chiếu
```

```
    //*****//
```

```
    /* Ta Quang Pho - 20215450 */
```

```
    int tmp = x;           // Tạo biến trung gian để lưu giá trị x
```

```
    x = y;                 // Gán giá trị x = giá trị y
```

```
    y = z;                 // Gán giá trị y = giá trị z
```

```
    z = tmp;               // Gán giá trị z = giá trị trung gian
```

```
                           // (= giá trị ban đầu của x)
```

```
    //*****//
```

```
}
```

```
int main() {
```

```
    int x, y, z;
```

```
    // # Nhập 3 số nguyên
```

```
    //*****//
```

```
    /* Ta Quang Pho - 20215450 */
```

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

//*****//

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

rotate(x, y, z);    // Gọi hàm rotate

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

return 0;

}
```

**Bài tập 2.3.** Viết chương trình yêu cầu nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị  $ax^2+bx+c$  với a, b, c định sẵn.

```

1  #include <iostream>
2
3  using namespace std;
4
5  /// Viết hàm get_value
6  //*****//
7  /* Ta Quang Pho - 20215450 */
8
9  int get_value(int x, int a = 2, int b = 1, int c = 0) {
10     return (a*x*x + b*x + c);    // Trả về  $ax^2+bx+c$ 
11 }
12
13 //*****//
14
15 int main(){
16     int x;
17     scanf("%d", &x);
18
19     int a = 2; /// giá trị mặc định của a
20     int b = 1; /// giá trị mặc định của b
21     int c = 0; /// giá trị mặc định của c
22
23     /// Nhập 3 số nguyên a, b, c từ bàn phím
24     //*****//
25     /* Ta Quang Pho - 20215450 */
26
27     scanf("%d%d%d", &a, &b, &c);
28
29     //*****//
30
31     // In ra giá trị  $2x^2 + x + 0$ 
32     printf("a=2, b=1, c=0: %d\n", get_value(x));
33
34     // In ra giá trị  $ax^2 + x + 0$ 
35     printf("a=%d, b=1, c=0: %d\n", a, get_value(x, a));
36
37     // In ra giá trị  $ax^2 + bx + 0$ 
38     printf("a=%d, b=%d, c=0: %d\n", a, b, get_value(x, a, b));
39
40     // In ra giá trị  $ax^2 + bx + c$ 
41     printf("a=%d, b=%d, c=%d: %d\n", a, b, c, get_value(x, a, b, c));
42
43     return 0;
44 }

```

	Input	Expected	Got	
✓	5 3 7 8	a=2, b=1, c=0: 55 a=3, b=1, c=0: 80 a=3, b=7, c=0: 110 a=3, b=7, c=8: 118	a=2, b=1, c=0: 55 a=3, b=1, c=0: 80 a=3, b=7, c=0: 110 a=3, b=7, c=8: 118	✓
✓	9 -1 5 -3	a=2, b=1, c=0: 171 a=-1, b=1, c=0: -72 a=-1, b=5, c=0: -36 a=-1, b=5, c=-3: -39	a=2, b=1, c=0: 171 a=-1, b=1, c=0: -72 a=-1, b=5, c=0: -36 a=-1, b=5, c=-3: -39	✓

Passed all tests! ✓



Mã nguồn:

```
#include <iostream>
```

```
using namespace std;
```

```
///# Viết hàm get_value
```

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

/\* Ta Quang Pho - 20215450 \*/

```
int get_value(int x, int a = 2, int b = 1, int c = 0) {
    return (a*x*x + b*x + c);    // Trả về  $ax^2+bx+c$ 
}
```

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

```
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
```

Tạ Quang Phô<sup>2</sup> - 20215450

```
//*****//
```

```
/* Ta Quang Pho - 20215450 */
```

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

```
//*****//
```

```
// In ra giá trị  $2x^2 + x + 0$ 
```

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

```
// In ra giá trị  $ax^2 + x + 0$ 
```

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

```
// In ra giá trị  $ax^2 + bx + 0$ 
```

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

```
// In ra giá trị  $ax^2 + bx + c$ 
```

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

```
return 0;
```

```
}
```

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

```

1  #include <iostream>
2
3  using namespace std;
4
5  int cube(int x) {
6      //# trả về lập phương của x
7      //*****//
8      /* Ta Quang Pho - 20215450 */
9
10     return (x*x*x);          // x^3
11
12     //*****//
13 }
14
15 //# viết hàm tính lập phương của một số kiểu double
16 double cube(double x) {
17     //# trả về lập phương của x
18     //*****//
19     /* Ta Quang Pho - 20215450 */
20
21     return (x*x*x);          // x^3
22
23     //*****//
24 }
25
26 }
27
28 int main() {
29     int n;
30     double f;
31     scanf("%d %lf", &n, &f);
32
33     // Hàm cube với tham số truyền vào kiểu int và trả về kết quả kiểu
34     // int được gọi
35     printf("Int: %d\n", cube(n));
36
37     // Hàm cube với tham số truyền vào kiểu double và trả về kết quả kiểu
38     // double được gọi
39     printf("Double: %.2lf\n", cube(f));
40
41     return 0;
42 }

```

	Input	Expected	Got	
✓	3 5.2	Int: 27 Double: 140.61	Int: 27 Double: 140.61	✓
✓	10 7.12	Int: 1000 Double: 360.94	Int: 1000 Double: 360.94	✓

Passed all tests! ✓

Mã nguồn:

```
#include <iostream>
```

```
using namespace std;
```

```
int cube(int x) {
```

```
    //# trả về lập phương của x
```

```
    //*****//
```

```
    /* Ta Quang Pho - 20215450 */
```

```
    return (x*x*x);        //  $x^3$ 
```

```
    //*****//
```

```
}
```

```
//# viết hàm tính lập phương của một số kiểu double
```

```
double cube(double x) {
```

```
    //# trả về lập phương của x
```

```
    //*****//
```

```
    /* Ta Quang Pho - 20215450 */
```

```
    return (x*x*x);        //  $x^3$ 
```

```
    //*****//
```

```
}
```

```
int main() {
```

```
    int n;
```

```
    double f;
```

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

```
    // Hàm cube với tham số truyền vào kiểu int và trả về kết quả kiểu
```

```
    // int được gọi
```

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

```
    // Hàm cube với tham số truyền vào kiểu double và trả về kết quả kiểu
```

```
    // double được gọi
```

Tạ Quang Phổ - 20215450

```
printf("Double: %.2lf\n", cube(f));  
  
return 0;  
}
```

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

```

1  #include <iostream>
2  #include <ostream>
3  #include <math.h>
4  #include <iomanip>
5
6  using namespace std;
7
8  struct Complex {
9      double real;
10     double imag;
11 };
12
13 Complex operator + (Complex a, Complex b) {
14     //*****//
15     /* Ta Quang Pho - 20215450 */
16
17     Complex tmp;
18     tmp.real = a.real + b.real; // Tổng phần thực
19     tmp.imag = a.imag + b.imag; // Tổng phần ảo
20     return tmp;
21
22     //*****//
23 }
24
25 Complex operator - (Complex a, Complex b) {
26     //*****//
27     /* Ta Quang Pho - 20215450 */
28
29     Complex tmp;
30     tmp.real = a.real - b.real; // Hiệu phần thực
31     tmp.imag = a.imag - b.imag; // Hiệu phần ảo
32     return tmp;
33
34     //*****//
35 }
36
37 Complex operator * (Complex a, Complex b) {
38     //*****//
39     /* Ta Quang Pho - 20215450 */
40
41     Complex tmp;
42     tmp.real = a.real * b.real - (a.imag * b.imag); // Phần thực của tích
43     tmp.imag = a.imag * b.real + b.imag * a.real; // Phần ảo của tích
44     return tmp;
45
46     //*****//
47 }
48
49 Complex operator / (Complex a, Complex b) {
50     //*****//
51     /* Ta Quang Pho - 20215450 */
52
53     Complex tmp;
54
55     // Công thức tính thương của 2 số phức bằng nhân liên hợp
56     tmp.real = (a.real * b.real + (a.imag * b.imag)) / (b.real * b.real + b.imag * b.imag);
57     tmp.imag = (a.imag * b.real - (a.real * b.imag)) / (b.real * b.real + b.imag * b.imag);
58     return tmp;
59
60     //*****//
61 }
62
63 // Đa năng hóa toán tử << để in ra số phức
64 ostream& operator << (ostream& out, const Complex &a) {
65     out << '(' << std::setprecision(2) << a.real << (a.imag >= 0 ? '+' : '-') << std::setpre
66     return out;

```

```

67 }
68
69 int main() {
70     double real_a, real_b, img_a, img_b;
71     cin >> real_a >> img_a;
72     cin >> real_b >> img_b;
73
74     Complex a{real_a, img_a}; // Khởi tạo biến phức a
75     Complex b{real_b, img_b}; // Khởi tạo biến phức b
76
77     cout << a << " + " << b << " = " << a + b << endl;
78     cout << a << " - " << b << " = " << a - b << endl;
79     cout << a << " * " << b << " = " << a * b << endl;
80     cout << a << " / " << b << " = " << a / b << endl;
81
82     return 0;
83 }

```

	Input	Expected	Got	
✓	3.2 4 1.1 -1	$(3.2+4i) + (1.1-1i) = (4.3+3i)$ $(3.2+4i) - (1.1-1i) = (2.1+5i)$ $(3.2+4i) * (1.1-1i) = (7.5+1.2i)$ $(3.2+4i) / (1.1-1i) = (-0.22+3.4i)$	$(3.2+4i) + (1.1-1i) = (4.3+3i)$ $(3.2+4i) - (1.1-1i) = (2.1+5i)$ $(3.2+4i) * (1.1-1i) = (7.5+1.2i)$ $(3.2+4i) / (1.1-1i) = (-0.22+3.4i)$	✓
✓	5.5 2 3 -1.5	$(5.5+2i) + (3-1.5i) = (8.5+0.5i)$ $(5.5+2i) - (3-1.5i) = (2.5+3.5i)$ $(5.5+2i) * (3-1.5i) = (20-2.2i)$ $(5.5+2i) / (3-1.5i) = (1.2+1.3i)$	$(5.5+2i) + (3-1.5i) = (8.5+0.5i)$ $(5.5+2i) - (3-1.5i) = (2.5+3.5i)$ $(5.5+2i) * (3-1.5i) = (20-2.2i)$ $(5.5+2i) / (3-1.5i) = (1.2+1.3i)$	✓

Passed all tests! ✓

Mã nguồn:

```

#include <iostream>
#include <ostream>
#include <math.h>
#include <iomanip>

```

using namespace std;

```

struct Complex {
    double real;
    double imag;
};

```

```

Complex operator + (Complex a, Complex b) {
    //*****//
    /* Ta Quang Pho - 20215450 */

```

```
Complex tmp;
tmp.real = a.real + b.real; // Tổng phần thực
tmp.imag = a.imag + b.imag; // Tổng phần ảo
return tmp;

//*****//
}

Complex operator - (Complex a, Complex b) {
//*****//
/* Ta Quang Pho - 20215450 */

Complex tmp;
tmp.real = a.real - b.real; // Hiệu phần thực
tmp.imag = a.imag - b.imag; // Hiệu phần ảo
return tmp;

//*****//
}

Complex operator * (Complex a, Complex b) {
//*****//
/* Ta Quang Pho - 20215450 */

Complex tmp;
tmp.real = a.real * b.real - (a.imag * b.imag); // Phần thực của tích
tmp.imag = a.imag * b.real + b.imag * a.real; // Phần ảo của tích
return tmp;

//*****//
}

Complex operator / (Complex a, Complex b) {
//*****//
/* Ta Quang Pho - 20215450 */

Complex tmp;

// Công thức tính thương của 2 số phức bằng nhân liên hợp
tmp.real = (a.real * b.real + (a.imag * b.imag)) / (b.real * b.real + b.imag * b.imag);
```



Tạ Quang Phổ - 20215450

```
tmp.imag = (a.imag * b.real - (a.real * b.imag)) / (b.real * b.real + b.imag * b.imag);
return tmp;

//*****//
}

// Đa năng hóa toán tử << để in ra số phức
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}; // Khởi tạo biến phức a
    Complex b{real_b, img_b}; // Khởi tạo biến phức 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 tập 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.

```

1 #include <iostream>
2
3 using namespace std;
4
5 void print(int n) {
6     printf("n=%d\n", n);
7 }
8
9 int mul3plus1(int n) {
10     return n * 3 + 1;
11 }
12
13 int div2(int n) {
14     return n / 2;
15 }
16
17 // khai báo các tham số cho các con trỏ hàm odd, even và output
18 void simulate(int n, int (*odd)(int n), int (*even)(int n), void (*output)(int n)) {
19     // Con trỏ tới hàm // Con trỏ tới hàm // Con trỏ tới hàm
20     // có tham số truyền// có tham số truyền// có tham số truyền
21     // vào kiểu int trả // vào kiểu int trả // vào kiểu int trả
22     // về kiểu int // về kiểu int // về kiểu void
23
24     (*output)(n); // Gọi hàm được trỏ bởi con trỏ output
25     if (n == 1) return;
26     if (n % 2 == 0) {
27         n = (*even)(n); // Nếu chẵn, gọi hàm được trỏ bởi con trỏ even
28     } else {
29         n = (*odd)(n); // Nếu lẻ, gọi hàm được trỏ bởi con trỏ odd
30     }
31     simulate(n, odd, even, output);
32 }
33
34 int main() {
35     int (*odd)(int) = NULL;
36     int (*even)(int) = NULL;
37
38     //*****//
39     /* Ta Quang Pho - 20215450 */
40
41     odd = &mul3plus1; // Gán địa chỉ hàm mul3plus1 cho odd
42     even = &div2; // Gán địa chỉ hàm div2 cho even
43
44     //*****//
45     /* Ta Quang Pho - 20215450 */
46
47     int n;
48     scanf("%d", &n);
49     simulate(n, odd, even, print);
50
51     return 0;
52 }

```

	Input	Expected	Got	
✓	19	n=19 n=58 n=29 n=88 n=44 n=22 n=11 n=34 n=17 n=52 n=26 n=13 n=40 n=20 n=10 n=5 n=16 n=8 n=4 n=2 n=1	n=19 n=58 n=29 n=88 n=44 n=22 n=11 n=34 n=17 n=52 n=26 n=13 n=40 n=20 n=10 n=5 n=16 n=8 n=4 n=2 n=1	✓
✓	33	n=33 n=100 n=50 n=25 n=76 n=38 n=19 n=58 n=29 n=88 n=44 n=22 n=11 n=34 n=17 n=52 n=26 n=13 n=40 n=20 n=10 n=5 n=16 n=8 n=4 n=2 n=1	n=33 n=100 n=50 n=25 n=76 n=38 n=19 n=58 n=29 n=88 n=44 n=22 n=11 n=34 n=17 n=52 n=26 n=13 n=40 n=20 n=10 n=5 n=16 n=8 n=4 n=2 n=1	✓

Passed all tests! ✓

Mã nguồn:

```
#include <iostream>

using namespace std;

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 số cho các con trỏ hàm odd, even và output
void simulate(int n, int (*odd)(int n), int (*even)(int n), void (*output)(int n)) {
    // Con trỏ tới hàm // Con trỏ tới hàm // Con trỏ tới hàm
    // có tham số truyền// có tham số truyền// có tham số truyền
    // vào kiểu int trả // vào kiểu int trả // vào kiểu int trả
    // về kiểu int // về kiểu int // về kiểu void

    (*output)(n); // Gọi hàm được trỏ bởi con trỏ output
    if (n == 1) return;
    if (n % 2 == 0) {
        n = (*even)(n); // Nếu chẵn, gọi hàm được trỏ bởi con trỏ even
    } else {
        n = (*odd)(n); // Nếu lẻ, gọi hàm được trỏ bởi con trỏ odd
    }
    simulate(n, odd, even, output);
}

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

    //*****//
    /* Ta Quang Pho - 20215450 */
```

Tạ Quang Phô - 20215450

```
odd = &mul3plus1;    // Gán địa chỉ hàm mul3plus1 cho odd
even = &div2;        // Gán địa chỉ hàm div2 cho even

//*****//
/* Ta Quang Pho - 20215450 */

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

return 0;
}
```

**Bài tập 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.

```

1  #include <iostream>
2
3  using namespace std;
4
5  //# viết hàm arr_sum
6  //*****//
7  /* Tạ Quang Phổ - 20215450 */
8
9  template <typename T>
10
11 // Hàm template để tính tổng các phần tử của hai mảng
12 // T: Kiểu dữ liệu của các phần tử trong mảng (ví dụ: int, double)
13 // Trả về: Tổng của tất cả phần tử trong cả hai mảng
14 T arr_sum(T* arr1, int size1, T* arr2, int size2) {
15     T sum = 0;
16
17     // Duyệt qua mảng đầu tiên và cộng dồn giá trị vào sum
18     for (int i = 0; i < size1; i++) {
19         sum += arr1[i];
20     }
21
22     // Duyệt qua mảng thứ hai và cộng dồn giá trị vào sum
23     for (int i = 0; i < size2; i++) {
24         sum += arr2[i];
25     }
26     return sum;
27 }
28
29 //*****//
30
31 int main() {
32     int val;
33     cin >> val;
34
35     // Tính toán với mảng các số nguyên
36     {
37         int a[] = {3, 2, 0, val};
38         int b[] = {5, 6, 1, 2, 7};
39         cout << arr_sum(a, 4, b, 5) << endl;
40     }
41
42     // Tính toán với mảng các số thực
43     {
44         double a[] = {3.0, 2, 0, val * 1.0};
45         double b[] = {5, 6.1, 1, 2.3, 7};
46         cout << arr_sum(a, 4, b, 5) << endl;
47     }
48
49     return 0;
50 }

```

	Input	Expected	Got	
✓	5	31 31.4	31 31.4	✓
✓	17	43 43.4	43 43.4	✓

Passed all tests! ✓

Mã nguồn:

```
#include <iostream>
```

```
using namespace std;
```

```
//# viết hàm arr_sum
```

```
//*****//
```

```
/* Ta Quang Pho - 20215450 */
```

```
template <typename T>
```

```
// Hàm template để tính tổng các phần tử của hai mảng
```

```
// T: Kiểu dữ liệu của các phần tử trong mảng (ví dụ: int, double)
```

```
// Trả về: Tổng của tất cả phần tử trong cả hai mảng
```

```
T arr_sum(T* arr1, int size1, T* arr2, int size2) {
```

```
    T sum = 0;
```

```
    // Duyệt qua mảng đầu tiên và cộng dồn giá trị vào sum
```

```
    for (int i = 0; i < size1; i++) {
```

```
        sum += arr1[i];
```

```
    }
```

```
    // Duyệt qua mảng thứ hai và cộng dồn giá trị vào sum
```

```
    for (int i = 0; i < size2; i++) {
```

```
        sum += arr2[i];
```

```
    }
```

```
    return sum;
```

```
}
```

```
//*****//
```

```
int main() {
```

```
    int val;
```

```
    cin >> val;
```

```
    // Tính toán với mảng các số nguyên
```

```
{
```

```
    int a[] = {3, 2, 0, val};
```

```
    int b[] = {5, 6, 1, 2, 7};
```

```
    cout << arr_sum(a, 4, b, 5) << endl;
```

```
}

// Tính toán với mảng các số thực
{
    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 tập 2.8.** Viết hàm so sánh cho thuật toán sắp xếp.

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <numeric>
5
6  using namespace std;
7
8  int main() {
9      int val1, val2;
10     cin >> val1 >> val2;
11
12     // Khởi tạo vector của các vector, mỗi vector con đại diện cho một mảng 1 chiều
13     vector< vector<int> > a = {
14         {1, 3, 7},
15         {2, 3, 4, val1}, // val1 là một giá trị nhập từ người dùng
16         {9, 8, 15},
17         {10, val2},      // val2 là một giá trị nhập từ người dùng
18     };
19
20     // Sắp xếp các vector con trong vector 'a' theo tổng giá trị của chúng giảm dần
21     //*****//
22     /* Ta Quang Pho - 20215450 */
23
24     // Sử dụng hàm sort với hàm so sánh nặc danh
25     sort(a.begin(), a.end(), [](const vector<int>& arr1, const vector<int>& arr2) {
26
27         int sum1 = accumulate(arr1.begin(), arr1.end(), 0);
28
29         int sum2 = accumulate(arr2.begin(), arr2.end(), 0);
30         // So sánh tổng hai vector để sắp xếp
31         return (sum1 > sum2); // Sắp xếp giảm dần
32     });
33
34     //*****//
35
36     for (const auto &v : a) {
37         for (int it : v) {
38             cout << it << ' ';
39         }
40         cout << endl;
41     }
42     return 0;
43 }

```

	Input	Expected	Got	
✓	-10 -5	9 8 15 1 3 7 10 -5 2 3 4 -10	9 8 15 1 3 7 10 -5 2 3 4 -10	✓
✓	100 -100	2 3 4 100 9 8 15 1 3 7 10 -100	2 3 4 100 9 8 15 1 3 7 10 -100	✓

Passed all tests! ✓

Mã nguồn:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>

using namespace std;

int main() {
    int val1, val2;
    cin >> val1 >> val2;

    // Khởi tạo vector của các vector, mỗi vector con đại diện cho một mảng 1 chiều
    vector< vector<int> > a = {
        {1, 3, 7},
        {2, 3, 4, val1}, // val1 là một giá trị nhập từ người dùng
        {9, 8, 15},
        {10, val2}, // val2 là một giá trị nhập từ người dùng
    };

    // Sắp xếp các vector con trong vector 'a' theo tổng giá trị của chúng giảm dần
    //*****//
    /* Ta Quang Pho - 20215450 */

    // Sử dụng hàm sort với hàm so sánh nặc danh
    sort(a.begin(), a.end(), [](const vector<int>& arr1, const vector<int>& arr2) {

        int sum1 = accumulate(arr1.begin(), arr1.end(), 0);

        int sum2 = accumulate(arr2.begin(), arr2.end(), 0);
        // So sánh tổng hai vector để sắp xếp
        return (sum1 > sum2); // Sắp xếp giảm dần
    });

    //*****//

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

**Bài tập 2.9:** 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<sup>-6</sup> và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản.

```

1  #include <vector>
2  #include <algorithm>
3  #include <cmath>
4  #include <ctime>
5  #include <cstdio>
6  #include <iostream>
7
8  using namespace std;
9
10 // Giới hạn đầu vào cho hàm sigmoid
11 const int LIMIT = 100;
12 // Số lượng giá trị tính trước để tối ưu hàm sigmoid
13 const int NUM_ITER = 100000;
14 // Tổng số lượng giá trị đầu vào để kiểm tra
15 const int NUM_INPUTS = NUM_ITER * 100;
16
17 // Hàm sigmoid chậm - tính theo công thức chuẩn
18 double sigmoid_slow(double x) {
19     return 1.0 / (1.0 + exp(-x));
20 }
21
22 // Mảng chứa các giá trị đầu vào để kiểm tra
23 double x[NUM_INPUTS];
24
25 // Hàm chuẩn bị các giá trị đầu vào ngẫu nhiên
26 void prepare_input() {
27     const int PRECISION = 1000000; // Độ chính xác cho việc sinh số
28     const double RANGE = LIMIT / 20.0;
29     for (int i = 0; i < NUM_INPUTS; ++i) {
30         x[i] = RANGE * (rand() % PRECISION - rand() % PRECISION) / PRECISION;
31     }
32 }
33
34 // Con trỏ toàn cục để trỏ tới vùng lưu trữ các giá trị sigmoid
35 // đã được tính trước
36 double *y;
37 // Khoảng cách giữa hai giá trị liên tiếp được tính trước
38 double delta = 2.0 * LIMIT / NUM_ITER;
39
40 // Hàm tính trước các giá trị của hàm sigmoid
41 void precalc() {
42     // Cấp phát bộ nhớ cho mảng y
43     y = new double[NUM_ITER];
44     // Tính toán và lưu trữ các giá trị sigmoid đã được tính trước
45     for (int i = 0; i < NUM_ITER; i++)
46         y[i] = sigmoid_slow(delta * i - LIMIT);
47 }
48

```

```

49 // Hàm sigmoid nhanh - sử dụng phép nội suy tuyến tính
50 double sigmoid_fast(double x) {
51     // Nếu x nằm ngoài phạm vi đã được tính trước, trả về giá trị biên
52     if (x < -LIMIT) return 0;
53     if (x > LIMIT) return 1;
54     // Tính chỉ số tương ứng trong mảng y
55     int n = (x + LIMIT) / delta;
56     // Tính giá trị sigmoid nhanh thông qua nội suy tuyến tính
57     return y[n] + (x + LIMIT - n * delta) * (y[n + 1] - y[n]) / delta;
58 }
59
60 // Hàm đo hiệu suất của hàm tính sigmoid
61 double benchmark(double (*calc)(double), vector<double> &result) {
62     const int NUM_TEST = 20; // SỐ TEST CASE
63
64     double taken = 0;
65     result = vector<double>();
66     result.reserve(NUM_ITER);
67
68     int input_id = 0;
69     clock_t start = clock();
70     for (int t = 0; t < NUM_TEST; ++t) {
71         double sum = 0;
72         for (int i = 0; i < NUM_ITER; ++i) {
73             double v = fabs(calc(x[input_id]));
74             sum += v;
75             if (t == 0) result.push_back(v);
76             if ((++input_id) == NUM_INPUTS) input_id = 0;
77         }
78     }
79     clock_t finish = clock();
80     taken = (double)(finish - start);
81     return taken;
82 }
83
84 // Hàm kiểm tra tính chính xác của hai hàm tính sigmoid
85 bool is_correct(const vector<double> &a, const vector<double> &b) {
86     const double EPS = 1e-6; // Sai số cho phép khi so sánh
87
88     if (a.size() != b.size()) return false;
89     for (int i = 0; i < (int)a.size(); ++i) {
90         if (fabs(a[i] - b[i]) > EPS) {
91             return false;
92         }
93     }
94     return true;
95 }
96
97 int main() {
98     // Chuẩn bị dữ liệu đầu vào
99     prepare_input();
100     // Tính toán trước các giá trị sigmoid
101     precalc();
102
103     vector<double> a, b;

```

Tạ Quang Phổ - 20215450

```
104 double slow = benchmark(sigmoid_slow, a);
105 double fast = benchmark(sigmoid_fast, b);
106
107 double xval;
108 scanf("%lf", &xval);
109 printf("%.2f \n", sigmoid_fast(xval));
110
111 if (is_correct(a, b) && (slow / fast > 1.3)) {
112     cout << "Correct answer! Your code is faster at least 30%" << endl;
113 } else {
114     printf("Wrong answer or your code is not fast enough!\n");
115 }
116
117 // Giải phóng bộ nhớ đã cấp phát cho y
118 delete[] y;
119
120 return 0;
121 }
```

Check

	Input	Expected	Got
✓	1.5	0.82 Correct answer! Your code is faster at least 30%!	0.82 Correct answer! Your code is faster at least 30%!
✓	2.15	0.90 Correct answer! Your code is faster at least 30%!	0.90 Correct answer! Your code is faster at least 30%!

Passed all tests! ✓

Mã nguồn :

```
#include <vector>
#include <algorithm>
#include <cmath>
#include <ctime>
#include <cstdio>
#include <iostream>
```

```
using namespace std;
```

```
// Giới hạn đầu vào cho hàm sigmoid
const int LIMIT = 100;
// Số lượng giá trị tính trước để tối ưu hàm sigmoid
```

IT3040 – 2023.1 – Mã lớp TH: 732830

```
const int NUM_ITER = 100000;
// Tổng số lượng giá trị đầu vào để kiểm tra
const int NUM_INPUTS = NUM_ITER * 100;

// Hàm sigmoid chậm - tính theo công thức chuẩn
double sigmoid_slow(double x) {
    return 1.0 / (1.0 + exp(-x));
}

// Mảng chứa các giá trị đầu vào để kiểm tra
double x[NUM_INPUTS];

// Hàm chuẩn bị các giá trị đầu vào ngẫu nhiên
void prepare_input() {
    const int PRECISION = 1000000; // Độ chính xác cho việc sinh số
    const double RANGE = LIMIT / 20.0;
    for (int i = 0; i < NUM_INPUTS; ++i) {
        x[i] = RANGE * (rand() % PRECISION - rand() % PRECISION) / PRECISION;
    }
}

// Con trỏ toàn cục để trỏ tới vùng lưu trữ các giá trị sigmoid
// đã được tính trước
double *y;
// Khoảng cách giữa hai giá trị liên tiếp được tính trước
double delta = 2.0 * LIMIT / NUM_ITER;

// Hàm tính trước các giá trị của hàm sigmoid
void precalc() {
    // Cấp phát bộ nhớ cho mảng y
    y = new double[NUM_ITER];
    // Tính toán và lưu trữ các giá trị sigmoid đã được tính trước
    for (int i = 0; i < NUM_ITER; i++)
        y[i] = sigmoid_slow(delta * i - LIMIT);
}

// Hàm sigmoid nhanh - sử dụng phép nội suy tuyến tính
double sigmoid_fast(double x) {
    // Nếu x nằm ngoài phạm vi đã được tính trước, trả về giá trị biên
    if (x < -LIMIT) return 0;
```

```

if (x > LIMIT) return 1;
// Tính chỉ số tương ứng trong mảng y
int n = (x + LIMIT) / delta;
// Tính giá trị sigmoid nhanh thông qua nội suy tuyến tính
return y[n] + (x + LIMIT - n * delta) * (y[n + 1] - y[n]) / delta;
}

// Hàm đo hiệu suất của hàm tính sigmoid
double benchmark(double (*calc)(double), vector<double> &result) {
    const int NUM_TEST = 20; // Số TEST CASE

    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;
}

// Hàm kiểm tra tính chính xác của hai hàm tính sigmoid
bool is_correct(const vector<double> &a, const vector<double> &b) {
    const double EPS = 1e-6; // Sai số cho phép khi so sánh

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

```

```
    }  
    return true;  
}  
  
int main() {  
    // Chuẩn bị dữ liệu đầu vào  
    prepare_input();  
    // Tính toán trước các giá trị sigmoid  
    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)) {  
        cout << "Correct answer! Your code is faster at least 30%!" << endl;  
    } else {  
        printf("Wrong answer or your code is not fast enough!\n");  
    }  
  
    // Giải phóng bộ nhớ đã cấp phát cho y  
    delete[] y;  
  
    return 0;  
}
```



**Bài tập 2.11:** 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  $N+M-1$ .

```

1  #include <bits/stdc++.h>
2
3  //*****//
4
5  /* Tạ Quang Phổ - 20215450 */
6
7  //*****//
8
9  using namespace std;
10
11 int n, m;           // n là bậc của đa thức A, m là bậc của đa thức B
12 int* a, *b, *result; // mảng a, b, result lưu lần lượt các phần tử của đa thức A, B, tích 2 đa thức
13
14 //Hàm khởi tạo các mảng và nhập dữ liệu đầu vào
15 void input() {
16     cin >> n;
17     a = new int [n + 1];           // Cấp phát vùng nhớ cho mảng A
18     memset(a, 0, (n + 1)*sizeof(int)); // Đặt giá trị các phần tử về 0 tránh vùng nhớ rác
19     for (int i = 0; i <= n; i++) {
20         cin >> a[i];
21     }
22
23     cin >> m;
24     b = new int [m + 1];           // Cấp phát vùng nhớ cho mảng B
25     memset(b, 0, (m + 1)*sizeof(int)); // Đặt giá trị các phần tử về 0 tránh vùng nhớ rác
26     for (int i = 0; i <= m; i++) {
27         cin >> b[i];
28     }
29
30     result = new int[n + m];       // Cấp phát vùng nhớ lưu tích 2 mảng
31     memset(result, 0, (n + m)*sizeof(int)); // Đặt giá trị các phần tử về 0 tránh vùng nhớ rác
32 }
33
34 // Hàm nhân 2 đa thức
35 void multiPoly() {
36     for (int i = 0; i <= n; i++) {
37         for (int j = 0; j <= m; j++) {
38             result[i + j] += a[i] * b[j]; // Cộng dồn các phần tử cùng bậc
39             result[i + j] += a[i] * b[j]; // Cộng dồn các phần tử cùng bậc
40         }
41     }
42
43     // Hàm in kết quả
44     void printResult() {
45         int tmp = 0;
46         int level = n + m;
47         for (int i = 0; i <= level; i++) {
48             tmp = tmp ^ result[i]; // tmp lưu giá trị XOR giữa tất cả các phần tử trong mảng
49         }
50
51         cout << tmp;
52     }
53
54     // Hàm thu hồi bộ nhớ đã cấp phát
55     void deleteAllocated() {
56         delete [] a;
57         delete [] b;
58         delete [] result;
59     }

```

```
61 int main () {
62     ios_base :: sync_with_stdio(false);
63     cin.tie(0); cout.tie(0);
64
65     input();
66     multiPoly();
67     printResult();
68     deleteAllocated();
69
70     cout << '\n' << "Ta Quang Pho - 20215450";
71
72     return 0;
73 }
74
```

Mã nguồn:

```
#include <bits/stdc++.h>
```

```
//*****//
```

```
/* Ta Quang Pho - 20215450 */
```

```
//*****//
```

```
using namespace std;
```

```
int n, m;           // n là bậc của đa thức A, m là bậc của đa thức B
```

```
int* a, *b, *result; // mảng a, b, result lưu lần lượt các phần tử của đa thức A, B, tích 2  
đa thức
```

```
//Hàm khởi tạo các mảng và nhập dữ liệu đầu vào
```

```
void input() {
```

```
    cin >> n;
```

```
    a = new int [n + 1];           // Cấp phát vùng nhớ cho mảng A
```

```
    memset(a, 0, (n + 1)*sizeof(int)); // Đặt giá trị các phần tử về 0 tránh vùng nhớ rác
```

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

```
        cin >> a[i];
```

```
    }
```

```
    cin >> m;
```

```
    b = new int [m + 1];           // Cấp phát vùng nhớ cho mảng B
```

```
    memset(b, 0, (m + 1)*sizeof(int)); // Đặt giá trị các phần tử về 0 tránh vùng nhớ rác
```

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

```
        cin >> b[i];
```

```
    }
```

```
    result = new int[n + m];           // Cấp phát vùng nhớ lưu tích 2 mảng
```

```
memset(result, 0, (n + m)*sizeof(int)); // Đặt giá trị các phần tử về 0 tránh vùng nhớ
rác
}
```

// Hàm nhân 2 đa thức

```
void multiPoly() {
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= m; j++) {
            result[i + j] += a[i] * b[j]; // Cộng dồn các phần tử cùng bậc
        }
    }
}
```

// Hàm in kết quả

```
void printResult() {
    int tmp = 0;
    int level = n + m;
    for (int i = 0; i <= level; i++) {
        tmp = tmp ^ result[i]; // tmp lưu giá trị XOR giữa tất cả các phần tử trong
        mảng
    }

    cout << tmp;
}
```

// Hàm thu hồi bộ nhớ đã cấp phát

```
void deleteAllocated() {
    delete [] a;
    delete [] b;
    delete [] result;
}
```

int main () {

```
    ios_base :: sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
```

```
    input();
    multiPoly();
    printResult();
    deleteAllocated();
```

## Tạ Quang Phổ - 20215450

```
cout << '\n' << "Ta Quang Pho - 20215450";

return 0;
}
```

### Case 1:

```
9 21 62 27 90 59 63 26 40 26 72

8 36 11 68 67 29 82 30 62 23
5196
-----
Process exited after 2.991 seconds with return value 0
Press any key to continue . . .
```

### Case 2:

```
99 67 35 29 2 22 58 69 67 93 56 11 42 29 73 21 19 84 37 98 24 15 70 13 26 91 80 56 73 62 70 96 81 5 25 84 27 36 5 46 29
13 57 24 95 82 45 14 67 34 64 43 50 87 8 76 78 88 84 3 51 54 99 32 60 76 68 39 12 26 86 94 39 95 70 34 78 67 1 97 2 17 9
2 52 56 1 80 86 41 65 89 44 19 40 29 31 17 97 71 81 75

98 9 27 67 56 97 53 86 65 6 83 19 24 28 71 32 29 3 19 70 68 8 15 40 49 96 23 18 45 46 51 21 55 79 88 64 28 41 50 93 0 34
64 24 14 87 56 43 91 27 65 59 36 32 51 37 28 75 7 74 21 58 95 29 37 35 93 18 28 43 11 28 29 76 4 43 63 13 38 6 40 4 18
28 88 69 17 17 96 24 43 70 83 90 99 72 25 44 90 5
125190
-----
Process exited after 1.592 seconds with return value 0
Press any key to continue . . .
```

### Case 3:

```
999 39 54 86 69 82 42 64 97 7 55 4 48 11 22 28 99 43 46 68 40 22 11 10 5 1 61 30 78 5 20 36 44 26 22 65 8 16 82 58 24 37
62 24 0 36 52 99 79 50 68 71 73 31 81 30 33 94 60 63 99 81 99 96 59 73 13 68 90 95 26 66 84 40 90 84 76 42 36 7 45 56 7
9 18 87 12 48 72 59 9 36 10 42 87 6 1 13 72 21 55 19 99 21 4 39 11 40 67 5 28 27 50 84 58 20 24 22 69 96 81 30 84 92 72
72 50 25 85 22 99 40 42 98 13 98 90 24 90 9 81 19 36 32 55 94 4 79 69 73 76 50 55 60 42 79 84 93 5 21 67 4 13 61 54 26 5
9 44 2 2 6 84 21 42 68 28 89 72 8 58 98 36 8 53 48 3 33 33 48 90 54 67 46 68 29 0 46 88 97 49 90 3 33 63 97 53 92 86 25
52 96 75 88 57 29 36 60 14 21 60 4 28 27 50 48 56 2 94 97 99 43 39 2 28 3 0 81 47 38 59 51 35 34 39 92 15 27 4 29 49 64
85 29 43 35 77 0 38 71 49 89 67 88 92 95 43 44 29 90 82 40 41 69 26 32 61 42 60 17 23 61 81 9 90 25 96 67 77 34 90 26 24
57 14 68 5 58 12 86 0 46 26 94 16 52 78 29 46 90 47 70 51 80 31 93 57 27 12 86 14 55 12 90 12 79 10 69 89 74 55 41 20 3
3 87 88 38 66 70 84 56 17 6 60 49 37 5 59 17 18 45 83 73 58 73 37 89 83 7 78 57 14 71 29 0 59 18 38 25 88 74 33 57 81 93
58 70 99 17 39 69 63 22 94 73 47 31 62 82 90 92 91 57 15 21 57 74 91 47 51 31 21 37 40 54 30 98 25 81 16 16 2 31 39 96
4 38 80 18 21 70 62 12 79 77 85 36 4 76 83 7 59 57 44 99 11 27 50 36 60 18 5 63 49 44 11 5 34 91 75 55 14 89 68 93 18 5
82 22 82 17 30 93 74 26 93 86 53 43 74 14 13 79 77 62 75 88 19 10 32 94 17 46 35 37 91 53 43 73 28 25 91 10 18 17 36 63
55 90 58 30 4 71 61 33 85 89 73 4 51 5 50 68 3 85 6 95 39 49 20 67 26 63 77 96 81 65 60 36 55 70 18 11 42 32 96 79 21 70
84 72 27 34 40 83 72 98 30 63 47 50 30 73 14 59 22 47 24 82 35 32 4 54 43 98 86 40 78 59 62 62 83 41 48 23 24 72 22 54
35 21 57 65 47 71 76 69 18 1 3 53 33 7 59 28 6 97 20 84 8 34 98 91 76 98 15 52 71 89 59 6 10 16 24 9 39 0 78 9 53 81 14
38 89 26 67 47 23 87 31 32 22 81 75 50 79 90 54 50 31 13 57 94 81 81 3 20 33 82 81 87 15 96 25 4 22 92 51 97 32 34 81 6
15 57 8 95 99 62 97 83 76 54 77 9 87 32 82 21 66 63 60 82 11 85 86 85 30 90 83 14 76 16 20 92 25 28 39 25 90 36 60 18 43
37 28 82 21 10 55 88 25 15 70 37 53 8 22 83 50 57 97 27 26 69 71 51 49 10 28 39 98 88 10 93 77 90 76 99 52 31 87 77 99
57 66 52 17 41 35 68 98 84 95 76 5 66 28 54 28 8 93 78 97 55 72 74 45 0 25 97 83 12 27 82 21 93 34 39 34 21 59 85 57 54
61 62 72 41 16 52 50 62 82 99 17 54 73 15 6 51 64 90 63 91 72 37 37 59 28 71 80 87 56 90 41 70 52 65 11 69 17 61 83 51 1
2 0 6 38 67 64 89 32 54 4 75 79 41 12 38 69 36 70 56 44 60 49 14 65 14 26 86 83 39 69 35 52 21 93 90 89 9 31 73 64 35 48
95 77 13 33 98 49 55 55 93 68 56

60 33 23 86 71 58 77 40 45 81 61 90 23 50 0 54 75 64 42 24 59 19 89 44 69 38 51 76 83 19 33 43 4 56 81 75 66 11 67 12 92
29 2 68 31 2 74 7 18 16 83 77 87 72 73 57 62 25 33 97 96 18 41 53 26 74 80 93 85 48 5 30 29 59 98 60 62 24 19 80 41 2 1
0 80 26 83 89 40 8 23 38 57 93 31 10 20 5 90 13 91 38 70 21 67 29 71 80 43 95 99 24 88 54 86 69 32 69 10 73 30 33 63 87
79 94

998 49 99 51 39 64 42 30 86 15 49 15 86 81 11 34 33 87 22 87 73 43 19 42 54 44 24 39 59 63 18 53 12 69 5 4 33 99 34 19 1
5 35 87 53 69 50 87 2 37 62 89 10 5 60 4 11 57 29 3 16 92 21 22 5 43 79 61 28 78 47 0 45 82 87 99 51 89 86 53 26 48 94 3
```

## Tạ Quang Phổ - 20215450

```
6 6 7 92 17 64 21 20 80 66 94 54 23 37 33 84 17 12 31 17 9 65 56 8 69 45 95 22 71 95 69 59 1 76 52 71 40 25 43 72 91 89
27 66 78 12 50 96 24 33 13 86 99 70 94 68 15 41 42 39 37 63 98 90 39 2 13 31 28 57 4 71 46 83 38 25 95 40 21 72 26 34 58
25 56 52 45 72 46 39 11 83 3 61 25 42 16 39 74 44 96 30 67 94 13 57 19 60 50 92 32 76 79 90 53 35 95 98 7 41 37 70 76 4
0 84 1 83 0 92 9 96 40 39 63 35 4 73 6 64 23 51 49 51 30 39 4 65 86 2 25 79 91 47 7 84 31 61 19 31 53 28 27 94 19 43 81
23 68 87 39 43 38 88 94 20 80 98 86 18 52 63 98 95 10 5 79 42 66 98 25 72 78 53 66 97 48 47 72 16 86 12 59 77 0 53 97 32
3 35 51 7 98 49 54 61 6 34 3 25 84 28 97 63 33 15 60 81 14 85 97 0 97 8 29 49 13 79 34 16 14 85 75 65 86 30 26 92 16 29
69 52 9 66 15 95 33 28 76 47 13 26 0 62 86 29 63 0 8 97 16 75 82 44 40 20 26 18 65 94 99 34 46 8 53 62 3 86 42 32 34 7
10 34 69 96 15 84 96 24 82 65 51 16 61 43 37 87 61 2 81 60 88 79 20 41 93 24 28 35 8 62 42 70 96 63 66 11 0 15 87 34 32
90 50 93 33 39 80 94 93 13 54 82 44 75 23 38 51 51 73 11 65 68 81 13 83 99 77 35 14 64 69 46 7 72 91 40 11 23 87 57 36 9
3 39 81 20 14 71 71 18 96 34 83 64 67 97 0 67 74 35 33 90 57 32 49 29 23 90 92 47 29 49 35 22 40 68 43 55 39 66 25 36 53
60 52 20 57 4 87 83 40 21 26 97 5 27 78 28 69 70 27 98 20 63 21 60 83 16 67 23 82 92 11 35 53 63 8 62 68 95 98 60 68 76
9 73 3 87 54 73 57 81 23 29 96 44 42 80 12 9 55 47 54 66 34 59 81 42 73 1 38 71 13 58 99 22 84 55 61 90 80 71 23 3 0 20
0 42 52 12 4 7 59 58 25 94 17 58 36 90 60 26 14 73 37 65 48 21 20 9 63 0 32 86 4 33 58 56 27 10 68 31 69 28 41 46 74 58
5 10 1 65 89 67 90 26 32 38 99 5 0 62 57 32 48 61 17 7 69 97 69 38 28 39 18 70 85 92 80 42 54 33 7 43 0 50 21 85 88 20
90 40 34 47 25 83 61 94 42 30 91 63 20 20 54 38 42 92 30 22 34 85 8 94 80 8 44 2 45 84 22 87 25 57 35 2 92 48 96 86 30 4
0 49 51 12 56 89 54 48 72 28 82 57 88 76 37 97 20 39 94 5 14 82 82 23 69 36 15 17 84 53 99 24 54 50 36 10 92 42 58 64 23
41 21 11 69 10 60 90 2 55 47 16 89 81 39 58 17 6 75 1 11 74 78 65 77 66 76 69 61 34 33 36 27 6 99 97 16 60 39 70 67 86
86 8 67 77 66 36 35 93 37 46 19 67 12 96 34 40 17 95 74 50 31 2 8 30 51 25 42 90 47 61 76 34 69 95 63 35 31 51 80 20 97
0 88 61 96 74 1 14 69 28 16 52 82 25 82 33 2 77 23 49 90 51 35 60 46 99 47 29 2 28 49 99 28 89 13 76 63 66 90 84 94 59 3
6 76 84 71 9 38 0 84 39 90 35 75 50 81 26 50 62 28 78 64 79 58 53 44 34 69 11 77 5 57 36 42 34 72 65 95 10 65 32 49 7 67
76 58 1 2 60 63 82 38 27 14 96 33 58 30 54 69 7 59 27 95 1 13 67 18 60 29 35 92 31 43 12 7 53 13 62 13 28 96 51 56 10 9
9 41 69 81 95 90 89 54 69 36 8 82 4 26 95 33 14 87 64 57 51 24 10 64 38 23 93 34 26 1 45 25 94 66 6 89 56 47 95 26 84 3
60 40 82 55 73 48 95 90
1146480
-----
Process exited after 5.753 seconds with return value 0
Press any key to continue . . . █
```

**Bài tập 2.12:** Hôm nay, cô giáo giao cho An một câu hỏi học 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.

```

1  #include <bits/stdc++.h>
2
3  //*****//
4
5  /* Ta Quang Pho - 20215450 */
6
7  //*****//
8
9  using namespace std;
10
11 int main () {
12     ios_base :: sync_with_stdio (false);
13     cin.tie(0); cout.tie(0);
14
15     vector <pair <int, int>> v;           // Vector v lưu các cặp {khóa, giá trị}
16
17     int key, value;                     // Các biến trung gian để đẩy giá trị vào vector
18     while (cin >> key) {                // Dừng khi gặp EOF
19         cin >> value;
20         v.push_back({key, value});
21     }
22
23     // Hàm sắp xếp với phép so sánh nặc danh
24     sort(v.begin(), v.end(), [](const pair <int, int> &a, const pair <int, int> &b) {
25         if (a.second == b.second) {    // Nếu 2 phần tử có giá trị bằng nhau
26             return a.first > b.first;  // thì ưu tiên khóa lớn hơn
27         }
28         return a.second > b.second;    // Giá trị lớn hơn đứng trước
29     });
30
31     // In ra các phần tử được sắp xếp theo yêu cầu
32     for (pair <int, int> item : v) {    // Duyệt các phần tử kiểu pair <int, int> trong vector
33         cout << item.first << ' ' << item.second << '\n'; // In (Khóa + ' ' + Giá trị)
34     }
35
36     return 0;
37 }
38

```

Mã nguồn:

```
#include <bits/stdc++.h>
```

```
//*****//
```

```
/* Ta Quang Pho - 20215450 */
```

```
//*****//
```

```
using namespace std;
```

```
int main () {
    ios_base :: sync_with_stdio (false);
    cin.tie(0); cout.tie(0);

    vector <pair <int, int>> v;          // Vector v lưu các cặp {khóa, giá trị}

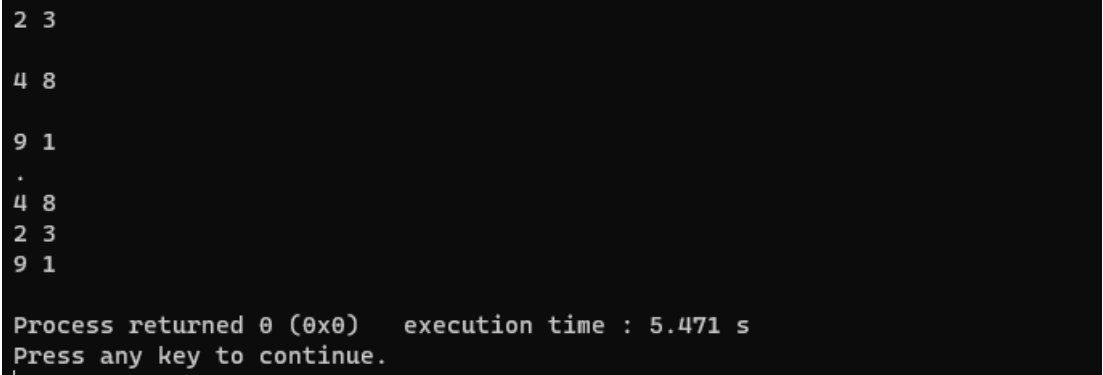
    int key, value;                     // Các biến trung gian để đẩy giá trị vào vector
    while (cin >> key) {                // Dừng khi gặp EOF
        cin >> value;
        v.push_back({key, value});
    }

    // Hàm sắp xếp với phép so sánh nặc danh
    sort(v.begin(), v.end(), [](const pair <int, int> &a, const pair <int, int> &b) {
        if (a.second == b.second) {    // Nếu 2 phần tử có giá trị bằng nhau
            return a.first > b.first;  // thì ưu tiên khóa lớn hơn
        }
        return a.second > b.second;    // Giá trị lớn hơn đứng trước
    });

    // In ra các phần tử được sắp xếp theo yêu cầu
    for (pair <int, int> item : v) {    // Duyệt các phần tử kiểu pair <int, int> trong vector
        cout << item.first << ' ' << item.second << "\n"; // In (Khóa + ' ' + Giá trị)
    }

    return 0;
}
```

Case 1:



```
2 3
4 8
9 1
1 4
8 2
3 9

Process returned 0 (0x0)   execution time : 5.471 s
Press any key to continue.
```

Case 2:

```
4 -8
-6 -6
2 6
-4 -6
7 -9
5 3
1 -2
1 -3
2 -1
-1 8
0 -3
-1 2
1 -8
3 8
10 -8
-9 0
-4 -4
-4 7
9 0
5 -2
-6 5
-10 -7
-4 0
9 2
6 5
5 -4
9 2
-4 -4
0 1
-5 -7
```

```
-3 -2
8 1
5 -9
9 -7
-2 -2
-6 0
-8 -1
-2 -3
-8 -1
9 1
2 6
-2 6
10 0
-3 3
2 -7
-10 -1
1 10
10 -6
-10 5
1 6
-1 -10
-9 -9
-1 -1
-5 6
2 -2
-6 -6
-2 -8
-8 -9
7 -9
10 -6
```



1	-5
8	-9
4	5
-5	-10
1	0
-7	9
5	-9
-10	-9
10	5
7	8
3	1
-2	-4
-7	-8
-2	6
-6	-10
5	-1
-1	-1
-10	-8
-9	-8
2	9
1	1
-6	-2
9	-3
-7	-9
2	10
3	10
2	-10
0	-10
5	-9
-2	-3

8	4
4	6
4	1
1	6
-8	-10
-7	-9
10	-1
-3	-6
2	0
-5	2
.	
3	10
2	10
1	10
2	9
-7	9
7	8
3	8
-1	8
-4	7
4	6
2	6
2	6
1	6
1	6
-2	6
-2	6
-5	6
10	5
6	5

```
4 5
-6 5
-10 5
8 4
5 3
-3 3
9 2
9 2
-1 2
-5 2
9 1
8 1
4 1
3 1
1 1
0 1
10 0
9 0
2 0
1 0
-4 0
-6 0
-9 0
10 -1
5 -1
2 -1
-1 -1
-1 -1
-8 -1
-8 -1
```

```
-10 -1
5 -2
2 -2
1 -2
-2 -2
-3 -2
-6 -2
9 -3
1 -3
0 -3
-2 -3
-2 -3
5 -4
-2 -4
-4 -4
-4 -4
1 -5
10 -6
10 -6
-3 -6
-4 -6
-6 -6
-6 -6
9 -7
2 -7
-5 -7
-10 -7
10 -8
4 -8
1 -8
```

```
-2 -8
-7 -8
-9 -8
-10 -8
8 -9
7 -9
7 -9
5 -9
5 -9
5 -9
-7 -9
-7 -9
-8 -9
-9 -9
-10 -9
2 -10
0 -10
-1 -10
-5 -10
-6 -10
-8 -10

Process returned 0 (0x0)   execution time : 7.559 s
Press any key to continue.
```

Case 3:

```
D:\C++\KTLT\B2\B2.12.exe  X + v
319262781 -223639388
879592302 -805221506
-250245742 -757176808
-948401744 863472684
80447005 -516478887
421465693 151620282
-917618581 -52498748
872453627 416684472
909042409 -949909022
-889680410 -557519485
.
-948401744 863472684
872453627 416684472
421465693 151620282
-917618581 -52498748
319262781 -223639388
80447005 -516478887
-889680410 -557519485
-250245742 -757176808
879592302 -805221506
909042409 -949909022

Process returned 0 (0x0)   execution time : 2.148 s
Press any key to continue.
```

Case 4:

## Tạ Quang Phô<sup>2</sup> - 20215450

```
D:\C++\KTLT\B2\B2.12.exe X + v
-621672744 -646412258
15794635 -2320603
475460025 -93534502
-152576442 944505201
-102158598 662482628
-996481340 -564919455
-270933211 974010350
39324943 -254809904
982057988 -155761188
-421160706 -349329142
-197322679 -777941303
956137904 300766341
377100342 -723096550
361321914 -903434347
223349857 810299144
-755324479 -484283156
622336131 148617796
-308222087 363739907
884415793 -417832051
-351128983 786469972
-136423368 940423129
-689596446 -631807300
-182059504 -452242229
891350039 -227966843
148532703 -447535204
-188219804 -895693870
-834525761 -548038846
-790375793 -840813355
775935340 543283981
-386414733 -339602373
888257394 -228317831
476819663 -122171558
-490858606 -673908899
529768322 -869883732
54988502 -760859518
-141073425 -465361047
-429040225 -749062923
250448111 -129648308
933253600 688733845
240896741 789097854
-405414749 -365057689
-722152611 403951883
529207869 -671331793
-73307506 -511194893
332251272 -916100188
-983079791 303605299
755171041 598308421
435429046 553079041
724311402 -456628521
705974678 967328920
209927371 -881613847
-665893261 -635399510
-41595183 897079538
572387910 75592658
577036603 381586746
787710450 -568271198
752086960 -211307975
-731765960 -910962208
694547477 209060971
753570008 -244428877
```

## Tạ Quang Phổ - 20215450

7858211	-74277265
-123465037	-8334743
-839961886	-692423154
-1116743	443317232
382741782	-62021563
563230615	758690457
-762397072	-699766393
-477608776	606773346
-728594226	-908930131
921340496	-776936940
-858927389	-530977273
463081312	-850304782
326631022	-340522427
-968012421	46448629
-962498351	196632617
778573387	-875466076
16969422	260055103
-484840673	-831494652
-12342979	-133582207
-652371896	787898891
-598378452	206631147
571675138	-112393966
507476909	785177604
882854768	761897692
-538482268	-374952121
564676075	850935842
167482100	881072243
115288731	275052442
403559643	338881513
122487885	-451061564
-980849931	-618663565
504687497	-596984619
-553585085	748644566
-555854455	-39183996
210125547	457521742
-331238921	-959637374
61226040	376935920
-250738126	-87119971
97204139	-222380549
504616457	41816496
.	
-270933211	974010350
705974678	967328920
-152576442	944505201
-136423368	940423129
-41595183	897079538
167482100	881072243
564676075	850935842
223349857	810299144
240896741	789097854
-652371896	787898891
-351128983	786469972
507476909	785177604
882854768	761897692
563230615	758690457
-553585085	748644566
933253600	688733845
-102158598	662482628
-477608776	606773346
755171041	598308421

## Tạ Quang Phổ - 20215450

```
210125547 457521742
-1116743 443317232
-722152611 403951883
577036603 381586746
61226040 376935920
-308222087 363739907
403559643 338881513
-983079791 303605299
956137904 300766341
115288731 275052442
16969422 260055103
694547477 209060971
-598378452 206631147
-962498351 196632617
622336131 148617796
572387910 75592658
-968012421 46448629
504616457 41816496
15794635 -2320603
-123465037 -8334743
-555854455 -39183996
382741782 -62021563
7858211 -74277265
-250738126 -87119971
475460025 -93534502
571675138 -112393966
476819663 -122171558
250448111 -129648308
-12342979 -133582207
982057988 -155761188
752086960 -211307975
97204139 -222380549
891350039 -227966843
888257394 -228317831
753570008 -244428877
39324943 -254809904
-386414733 -339602373
326631022 -340522427
-421160706 -349329142
-405414749 -365057689
-538482268 -374952121
884415793 -417832051
148532703 -447535204
122487885 -451061564
-182059504 -452242229
724311402 -456628521
-141073425 -465361047
-755324479 -484283156
-73307506 -511194893
-858927389 -530977273
-834525761 -548038846
-996481340 -564919455
787710450 -568271198
504687497 -596984619
-980849931 -618663565
-689596446 -631807300
-665893261 -635399510
-621672744 -646412258
529207869 -671331793
-490858606 -673908899
```

```
-839961886 -692423154
-762397072 -699766393
377100342 -723096550
-429040225 -749062923
54988502 -760859518
921340496 -776936940
-197322679 -777941303
-484840673 -831494652
-790375793 -840813355
463081312 -850304782
529768322 -869883732
778573387 -875466076
209927371 -881613847
-188219804 -895693870
361321914 -903434347
-728594226 -908930131
-731765960 -910962208
332251272 -916100188
-331238921 -959637374
```

```
Process returned 0 (0x0)   execution time : 2.285 s
Press any key to continue.
```

**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.

```
1  #include <bits/stdc++.h>
2  #define N 101
3
4  //*****//
5
6  /* Ta Quang Pho - 20215450 */
7
8  //*****//
9
10 using namespace std;
11
12 struct bigNum {
13     char sign; // Lưu dấu của số lớn (1 : dương, 0 : âm)
14     char num[N]; // Mảng lưu số lớn
15
16     // Hàm khởi tạo mặc định (giá trị = 0)
17     bigNum() {
18         sign = '1'; // Mặc định dấu dương
19         for (int i = 0; i < N; i++)
20             num[i] = '0'; // Số lớn = 0
21     }
22
23     // Khởi tạo số lớn từ 1 string thứ tự ngược
24     bigNum(string& str) {
25         int len = str.length(); // Lấy độ dài chuỗi
26         for (int i = N - 1; i >= N - len + 1; --i) {
27             num[i] = str[len - N + i]; // Lưu theo thứ tự ngược
28         }
29         for (int i = 0; i <= N - len; i++) {
30             num[i] = '0'; // Các phần tử còn lại = 0
31         }
32         sign = str[0]; // Dấu là phần tử đầu tiên của mảng
33     }
34
35     // Hàm nhân 10 (sử dụng cho việc dịch phép nhân)
36     void multiply10(bigNum& val, int level) {
37         for (int i = 0; i < N - level; i++) {
38             val.num[i] = val.num[i + level]; // Dịch sang trái level phần tử
39         }
40         for (int i = N - level; i < N; i++) {
41             val.num[i] = '0'; // Còn lại = 0
42         }
43     }
44 }
```



```

45 // Đa năng hóa toán tử + cho cấu trúc bigNum
46 bigNum operator+(const bigNum& operand) const {
47     bigNum result; // Tạo giá trị result = 0
48     int carry = 0; // Biến nhớ
49     for (int i = N - 1; i >= 0; --i) {
50         int sum = (num[i] - '0') + (operand.num[i] - '0') + carry;
51         // num là mảng lưu giá trị toán hạng thứ 1
52         // operand là toán hạng thứ 2
53         result.num[i] = sum % 10 + '0'; // Lưu kết quả
54         carry = sum / 10; // Tăng biến nhớ
55     }
56     return result;
57 }
58
59 // Đa năng hóa toán tử - cho cấu trúc bigNum
60 bigNum operator-(const bigNum& operand) const {
61     bigNum result; // Tạo giá trị result = 0
62     int borrow = 0; // Biến nhớ
63     for (int i = N - 1; i >= 0; --i) {
64         int x = num[i] - '0', y = operand.num[i] - '0';
65         // num là mảng lưu giá trị toán hạng thứ 1
66         // operand là toán hạng thứ 2
67         if (x >= y + borrow) { // Nếu số bị trừ > số trừ + nhớ => trừ bình thường
68             result.num[i] = x - (y + borrow) + '0';
69             borrow = 0;
70         } else {
71             result.num[i] = (x + 10) - (y + borrow) + '0';
72             // Ngược lại, tăng số bị trừ thêm 10 và tăng biến nhớ thêm 1
73             borrow = 1;
74         }
75     }
76     return result;
77 }
78
79 // Đa năng hóa toán tử * với số nguyên dương
80 bigNum operator*(int y) const {
81     bigNum result; // Tạo giá trị result = 0
82     result.sign = sign; // Dấu không đổi
83     int carry = 0; // Biến nhớ
84     for (int i = N - 1; i >= 0; --i) {
85         int x = num[i] - '0'; // num là mảng lưu giá trị toán hạng
86         int part = x * y + carry; // y là số nguyên cần nhân thêm
87         result.num[i] = part % 10 + '0';
88         carry = part / 10;
89     }
90     return result;
91 }
92
93 // Đa năng hóa toán tử * với 1 bigNum
94 bigNum operator*(const bigNum& operand) {
95     bigNum result; // Tạo giá trị result = 0;
96     for (int i = N - 1; i >= 0; --i) {
97         int x = operand.num[i] - '0';
98         bigNum temp = (*this) * x; // nhân số hiện tại với x
99         multiply10(temp, N - i - 1); // dịch trái N - i - 1 đơn vị
100         result = result + temp; // Cộng dồn số lớn vào kết quả
101     }
102     result.sign = '1' - ((sign - '0') ^ (operand.sign - '0')) + '0'; // Xét dấu của kết quả
103     return result;
104 }
105

```

```

106 // Đa năng hóa toán tử > với bigNum
107 bool operator>(const bigNum& operand) const {
108     for (int i = 0; i < N; i++) { // duyệt từ hàng cao -> thấp (VD từ trăm -> chục -> đơn vị)
109         if (num[i] == operand.num[i])
110             continue;
111         else if (num[i] > operand.num[i]) // trả về true ngay khi tìm được phần tử đầu tiên lớn hơn
112             return true;
113         else
114             return false;
115     }
116     return false;
117 }
118 };
119
120 // Ghi đè toán tử << cho việc in dữ liệu
121 ostream& operator<<(ostream& stream, const bigNum& val) { // Làm việc với ostream (luồng output)
122     stream << val.sign; // In ra dấu
123     int i = 0;
124     while (i < N && val.num[i] == '0') // Vòng lặp để tăng i đến chỗ cần tăng giá trị
125         i++;
126     for (i; i < N; i++) {
127         stream << val.num[i]; // In ra từng ký tự trong số lớn
128     }
129     return stream; // Trả về luồng output
130 }
131
132 // Overloading the input stream operator for bigNum
133 istream& operator>>(istream& stream, bigNum& val) { // Làm việc với istream (luồng input)
134     string temp;
135     stream >> temp; // Nhập vào 1 string
136     val = bigNum(temp); // Khởi tạo bigNum với xâu temp được truyền vào
137     return stream; // Trả về luồng input
138 }
139
140 int main () {
141     ios_base::sync_with_stdio (false);
142     cin.tie(0); cout.tie(0);
143
144     bigNum n, m;
145     cin >> n;
146     cin >> m;
147     bigNum a = n * m, b = n * 3, c = m * 4, result;
148
149     // Kiểm tra dấu để có được result = n*m - 3*n
150     if (a.sign == '1') {
151         if (b.sign == '1') {
152             if (a > b) {
153                 result = a - b;
154             } else {
155                 result = b - a;
156                 a.sign = '0';
157             }
158         } else {
159             result = a + b;
160         }
161     } else {
162         if (b.sign == '1') {
163             result = a + b;
164             result.sign = '0';
165         } else {
166             if (a > b) {
167                 result = a - b;
168                 result.sign = '0';
169             } else {
170                 result = b - a;
171             }
172         }
173     }

```

```

172     }
173 }
174
175 // Kiểm tra dấu để có được result = n*m - 3*n + 4*m
176 if (result.sign == '1') {
177     if (c.sign == '1') {
178         result = result + c;
179     } else {
180         if (result > c) {
181             result = result - c;
182         } else {
183             result = c - result;
184             result.sign = '0';
185         }
186     }
187 } else {
188     if (c.sign == '0') {
189         result = result + c;
190         result.sign = '0';
191     } else {
192         if (c > result) {
193             result = c - result;
194         } else {
195             result = result - c;
196             result.sign = '0';
197         }
198     }
199 }
200
201 cout << result;
202
203 return 0;
204 }
205

```

Mã nguồn

```
#include <bits/stdc++.h>
```

```
#define N 101
```

```
/**/
```

```
/* Tạ Quang Phổ - 20215450 */
```

```
/**/
```

```
using namespace std;
```

```
struct bigNum {
```

```
    char sign; // Lưu dấu của số lớn (1 : dương, 0 : âm)
```

```
    char num[N]; // Mảng lưu số lớn
```

```
    // Hàm khởi tạo mặc định (giá trị = 0)
```

```

bigNum() {
    sign = '1'; // Mặc định dấu dương
    for (int i = 0; i < N; i++)
        num[i] = '0'; // Số lớn = 0
}

// Khởi tạo số lớn từ 1 string thứ tự ngược
bigNum(string& str) {
    int len = str.length(); // Lấy độ dài xâu
    for (int i = N - 1; i >= N - len + 1; --i) {
        num[i] = str[len - N + i]; // Lưu theo thứ tự ngược
    }
    for (int i = 0; i <= N - len; i++) {
        num[i] = '0'; // Các phần tử còn lại = 0
    }
    sign = str[0]; // Dấu là phần tử đầu tiên của mảng
}

// Hàm nhân 10 (sử dụng cho việc dịch phép nhân)
void multiply10(bigNum& val, int level) {
    for (int i = 0; i < N - level; i++) {
        val.num[i] = val.num[i + level]; // Dịch sang trái level phần tử
    }
    for (int i = N - level; i < N; i++) {
        val.num[i] = '0'; // Còn lại = 0
    }
}

// Đa năng hóa toán tử + cho cấu trúc bigNum
bigNum operator+(const bigNum& operand) const {
    bigNum result; // Tạo giá trị result = 0
    int carry = 0; // Biến nhớ
    for (int i = N - 1; i >= 0; --i) {
        int sum = (num[i] - '0') + (operand.num[i] - '0') + carry;
        // num là mảng lưu giá trị toán hạng thứ 1
        // operand là toán hạng thứ 2
        result.num[i] = sum % 10 + '0'; // Lưu kết quả
        carry = sum / 10; // Tăng biến nhớ
    }
    return result;
}

```

}

// Đa năng hóa toán tử - cho cấu trúc bigNum

bigNum operator-(const bigNum&amp; operand) const {

bigNum result; // Tạo giá trị result = 0

int borrow = 0; // Biến nhớ

for (int i = N - 1; i &gt;= 0; --i) {

int x = num[i] - '0', y = operand.num[i] - '0';

// num là mảng lưu giá trị toán hạng thứ 1

// operand là toán hạng thứ 2

if (x &gt;= y + borrow) { // Nếu số bị trừ &gt; số trừ + nhớ =&gt; trừ bình thường

result.num[i] = x - (y + borrow) + '0';

borrow = 0;

} else {

result.num[i] = (x + 10) - (y + borrow) + '0';

// Ngược lại, tăng số bị trừ thêm 10 và tăng biến nhớ thêm 1

borrow = 1;

}

}

return result;

}

// Đa năng hóa toán tử \* với số nguyên dương

bigNum operator\*(int y) const {

bigNum result; // Tạo giá trị result = 0

result.sign = sign; // Dấu không đổi

int carry = 0; // Biến nhớ

for (int i = N - 1; i &gt;= 0; --i) {

int x = num[i] - '0'; // num là mảng lưu giá trị toán hạng

int part = x \* y + carry; // y là số nguyên cần nhân thêm

result.num[i] = part % 10 + '0';

carry = part / 10;

}

return result;

}

// Đa năng hóa toán tử \* với 1 bigNum

bigNum operator\*(const bigNum&amp; operand) {

bigNum result; // Tạo giá trị result = 0;

for (int i = N - 1; i &gt;= 0; --i) {

Tạ Quang Phổ - 20215450

```
int x = operand.num[i] - '0';
bigNum temp = (*this) * x; // nhân số hiện tại với x
multiply10(temp, N - i - 1); // dịch trái N - i - 1 đơn vị
result = result + temp; // Cộng dồn số lớn vào kết quả
}
result.sign = '1' - ((sign - '0') ^ (operand.sign - '0')) + '0'; // Xét dấu của kết quả
return result;
}

// Đa năng hóa toán tử > với bigNum
bool operator>(const bigNum& operand) const {
    for (int i = 0; i < N; i++) { // duyệt từ hàng cao -> thấp (VD từ trăm -> chục ->
đơn vị)
        if (num[i] == operand.num[i])
            continue;
        else if (num[i] > operand.num[i]) // trả về true ngay khi tìm được phần tử đầu
tiên lớn hơn
            return true;
        else
            return false;
    }
    return false;
}
};

// Ghi đè toán tử << cho việc in dữ liệu
ostream& operator<<(ostream& stream, const bigNum& val) { // Làm việc với ostream
(luồng output)
    stream << val.sign; // In ra dấu
    int i = 0;
    while (i < N && val.num[i] == '0') // Vòng lặp để tăng i đến chỗ cần tăng giá trị
        i++;
    for (i; i < N; i++) {
        stream << val.num[i]; // In ra từng ký tự trong số lớn
    }
    return stream; // Trả về luồng output
}

// Overloading the input stream operator for bigNum
```

```
istream& operator>>(istream& stream, bigNum& val) {    // Làm việc với istream (luồng
input)
    string temp;
    stream >> temp; // Nhập vào 1 string
    val = bigNum(temp); // Khởi tạo bigNum với xâu temp được truyền vào
    return stream; // Trả về luồng input
}
```

```
int main () {
    ios_base :: sync_with_stdio (false);
    cin.tie(0); cout.tie(0);

    bigNum n, m;
    cin >> n;
    cin >> m;
    bigNum a = n * m, b = n * 3, c = m * 4, result;

    // Kiểm tra dấu để có được result = n*m - 3*n
    if (a.sign == '1') {
        if (b.sign == '1') {
            if (a > b) {
                result = a - b;
            } else {
                result = b - a;
                a.sign = '0';
            }
        } else {
            result = a + b;
        }
    } else {
        if (b.sign == '1') {
            result = a + b;
            result.sign = '0';
        } else {
            if (a > b) {
                result = a - b;
                result.sign = '0';
            } else {
                result = b - a;
            }
        }
    }
}
```

```
    }  
}  
  
// Kiểm tra dấu để có được result = n*m - 3*n + 4*m  
if (result.sign == '1') {  
    if (c.sign == '1') {  
        result = result + c;  
    } else {  
        if (result > c) {  
            result = result - c;  
        } else {  
            result = c - result;  
            result.sign = '0';  
        }  
    }  
} else {  
    if (c.sign == '0') {  
        result = result + c;  
        result.sign = '0';  
    } else {  
        if (c > result) {  
            result = c - result;  
        } else {  
            result = result - c;  
            result.sign = '0';  
        }  
    }  
}  
  
cout << result;  
  
return 0;  
}
```



Test case:

Case 1:

```
0121807015

1347227347
042294724910108772

Process returned 0 (0x0)   execution time : 1.401 s
Press any key to continue.
```

Case 2:

```
0800547253714

0389013676936
1311423830729145609193702

Process returned 0 (0x0)   execution time : 1.447 s
Press any key to continue.
```

Case 3:

```
1562862701008461237669505996967049208066942997054417216894858422269

1995808870599753186889299576925652087713749800604075758946132
156050367059396234423341580242182452120324069095999461007353885451042021791868583422263169793739911208078001014076206214
0731229

Process returned 0 (0x0)   execution time : 1.537 s
Press any key to continue.
```

Case 4:

```
12335657506361583062448207352982257795957130202802668419171402333

0646136594804541606956409923002364617388546944061622123259
015091537877901402558863232700453674335664686358003691913838753612198590816408724208321336468191584104245029667160088632
82

Process returned 0 (0x0)   execution time : 1.361 s
Press any key to continue.
```