

## Contents

Bài thực hành số 2 – Tuần 34.....	2
<b>Bài tập 2.1.</b> Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông.....	2
<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. ....	4
<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. ....	7
<b>Bài tập 2.4.</b> Viết các hàm tính lập phương của số nguyên và số thực. ....	10
<b>Bài tập 2.5.</b> Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức ....	13
<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 $n = 1$ . ....	17
<b>Bài tập 2.7.</b> Viết hàm tính tổng các phần tử trong hai mảng. ....	21
<b>Bài tập 2.8.</b> Viết hàm so sánh cho thuật toán sắp xếp. ....	24
<b>Bài tập 2.9.</b> Viết chương trình in ra tất cả các dãy con của một dãy cho trước. ....	27
<b>Bài tập 2.10.</b> Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ $N \times N$ theo công thức trực tiếp. ....	35
<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 $N+M-1$ .....	42
<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. ....	49
<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: .....	54

**Nguyễn Văn Duy – 20215334**  
**Bài thực hành số 2 – Tuần 34**

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

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

For example:

Dữ liệu đầu vào	Kết quả
3 4	$z = 5.00$

Đáp án: (penalty regime: 10, 20, ... %)

```
1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.1. Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông.
4 */
5 #include <stdio.h>
6 #include <math.h>
7 float get_hypotenuse(float x, float y) {
8     /*
9      * YOUR CODE HERE *
10     */
11     return sqrtf(x*x + y*y);
12 }
13 int main(){
14     float x, y;
15     scanf("%f%f", &x, &y);
16
17     float z = get_hypotenuse(x, y);
18     printf("z = %.2f\n", z);
19
20     return 0;
21 }
22 // Nguyen Van Duy - 20215334
23
```

Chấm thử

Chấm điểm

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình	
✓	3 4	$z = 5.00$	$z = 5.00$	✓
✓	5 6	$z = 7.81$	$z = 7.81$	✓

Hoàn thành được tất cả các bộ mẫu ✓

Trang tiếp

```
// Nguyen Van Duy - 20215334
```

```
/*
```

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

```
*/
```

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

```
#include <math.h>
```

```
float get_hypotenuse(float x, float y) {
```

```
    /*
```

```
# YOUR CODE HERE #  
  
*****/  
  
return sqrtf(x*x + y*y);  
}  
  
int main(){  
    float x, y;  
    scanf("%f%f", &x, &y);  
  
    float z = get_hypotenuse(x, y);  
    printf("z = %.2f\n", z);  
  
    return 0;  
}  
  
// Nguyen Van Duy - 20215334
```

**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.

**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.

For example:

Dữ liệu đầu vào	Kết quả
3 4 5	Before: 3, 4, 5 After: 4, 5, 3

Đáp án: (penalty regime: 10, 20, ... %)

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.2. Viết hàm hoán vị vòng tròn 3 biến a, b, c.
4 Sau khi thực hiện hàm, các biến a, b, c tương ứng
5 nhận các giá trị mới b, c, a.
6 */
7 #include <stdio.h>
8 void rotate(int &x, int &y, int &z) {
9     /*
10     # YOUR CODE HERE #
11     */
12     int temp = x;
13     x = y;
14     y = z;
15     z = temp;
16 }
17
18 int main() {
19     int x, y, z;
20
21     /* Nhập 3 số nguyên
22     */
23
24     # YOUR CODE HERE #
25     /*
26     scanf("%d%d%d", &x, &y, &z);
27
28     printf("Before: %d, %d, %d\n", x, y, z);
29     rotate(x, y, z);
30     printf("After: %d, %d, %d\n", x, y, z);
31
32     return 0;
33 }
34 // Nguyen Van Duy - 20215334

```

Chấm thử

Chấm điểm

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình	
✓	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	✓

Hoàn thành được tất cả các bộ mẫu ✓

Trang tiếp

// Nguyen Van Duy - 20215334

/\*

Bài 2.2. Viết hàm hoán vị vòng tròn 3 biến a, b, c.

Sau khi thực hiện hàm, các biến a, b, c tương ứng nhận các giá trị mới b, c, a.

```
*/  
  
#include <stdio.h>  
  
void rotate(int &x, int &y, int &z) {  
    /******  
    # YOUR CODE HERE #  
    *****/  
  
    int temp = x;  
  
    x = y;  
  
    y = z;  
  
    z = temp;  
}  
  
int main() {  
    int x, y, z;  
  
    //# Nhập 3 số nguyên  
    /******  
    # YOUR CODE HERE #  
    *****/  
  
    scanf("%d%d%d", &x, &y, &z);  
  
    printf("Before: %d, %d, %d\n", x, y, z);  
    rotate(x, y, z);  
    printf("After: %d, %d, %d\n", x, y, z);
```

```
    return 0;  
}  
// Nguyen Van Duy - 20215334
```

**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.

**Bài 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.

For example:

Dữ liệu đầu vào	Kết quả
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

Đáp án: (penalty regime: 10, 20, ... %)

```

1 // Nguyen Van Duy - 20215334
2 /*
3  Bài 2.3. Viết chương trình yêu cầu nhập giá trị
4  cho số nguyên x nhỏ hơn 100. In ra giá trị  $ax^2 + bx + c$ 
5  với a, b, c định sẵn.
6  */
7 #include <stdio.h>
8 // Viết hàm get_value
9 // *****
10 # YOUR CODE HERE #
11 // *****
12 int get_value(int x, int a = 2, int b = 1, int c = 0) {
13     return a*x*x + b*x + c;
14 }
15
16 int main(){
17     int x;
18     scanf("%d", &x);
19
20     int a = 2; // giá trị mặc định của a
21     int b = 1; // giá trị mặc định của b
22     int c = 0; // giá trị mặc định của c
23
24     // Nhập 3 số nguyên a, b, c từ bàn phím
25     // *****
26     # YOUR CODE HERE #
27     // *****
28     scanf("%d%d%d", &a, &b, &c);
29
30     printf("a=2, b=1, c=0: %d\n", get_value(x));
31     printf("a=%d, b=1, c=0: %d\n", a, get_value(x, a));
32     printf("a=%d, b=%d, c=0: %d\n", a, b, get_value(x, a, b));
33     printf("a=%d, b=%d, c=%d: %d\n", a, b, c, get_value(x, a, b, c));
34
35     return 0;
36 }
37 // Nguyen Van Duy - 20215334
38

```

Chấm thử

Chấm điểm

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình	
✓	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	✓

Hoàn thành được tất cả các bộ mẫu ✓

Trang tiếp

```
// Nguyen Van Duy - 20215334

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

#include <stdio.h>

//# Viết hàm get_value
/*****

# YOUR CODE HERE #
*****/

int get_value(int x, int a = 2, int b = 1, int c = 0) {
    return a*x*x + b*x + 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
    /*****

    # YOUR CODE HERE #
    *****/
```



\*\*\*\*\*/

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

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

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

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

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

```
return 0;
```

```
}
```

```
// Nguyen Van Duy - 20215334
```

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

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

For example:

Dữ liệu đầu vào	Kết quả
3 5.2	Int: 27 Double: 140.61

**Đáp án:** (penalty regime: 10, 20, ... %)

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.4. Viết các hàm tính lập phương của số nguyên và số thực.
4 */
5 #include <stdio.h>
6 int cube(int x) {
7     /* trả về lập phương của x
8     *****
9     # YOUR CODE HERE #
10    *****
11    return x*x*x;
12 }
13
14 /*# viết hàm tính lập phương của một số kiểu double
15 *****
16 # YOUR CODE HERE #
17 *****
18 double cube(double x) {
19     return x*x*x;
20 }
21
22 int main() {
23
24     int n;
25     double f;
26     scanf("%d %lf", &n, &f);
27
28     printf("Int: %d\n", cube(n));
29     printf("Double: %.21f\n", cube(f));
30
31     return 0;
32 }
33 // Nguyen Van Duy - 20215334

```

Chấm thử

Chấm điểm

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình	
✓	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	✓

Hoàn thành được tất cả các bộ mẫu ✓

Trang tiếp

// Nguyen Van Duy - 20215334

/\*

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

\*/

```
#include <stdio.h>

int cube(int x) {
    //# trả về lập phương của x
    /*****

    # YOUR CODE HERE #

    *****/

    return x*x*x;
}

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

# YOUR CODE HERE #

*****/

double cube(double x) {
    return x*x*x;
}

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

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

    return 0;
}
```

// Nguyen Van Duy - 20215334

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

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

For example:

Dữ liệu đầu vào	Kết quả
3.2 4	$(3.2+4i) + (1.1-1i) = (4.3+3i)$
1.1 -1	$(3.2+4i) - (1.1-1i) = (2.1+5i)$
	$(3.2+4i) * (1.1-1i) = (7.5+1.2i)$
	$(3.2+4i) / (1.1-1i) = (-0.22+3.4i)$

Đáp án: (penalty regime: 10, 20, ... %)

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.5. Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức.
4 */
5 #include <iostream>
6 #include <ostream>
7 #include <math.h>
8 #include <iomanip>
9 using namespace std;
10 struct Complex {
11     double real;
12     double imag;
13 };
14
15 Complex operator + (Complex a, Complex b) {
16     /*
17     # YOUR CODE HERE #
18     */
19     return Complex {a.real + b.real, a.imag + b.imag};
20 }
21
22 Complex operator - (Complex a, Complex b) {
23     /*
24     # YOUR CODE HERE #
25     */
26     return Complex {a.real - b.real, a.imag - b.imag};
27 }
28
29 Complex operator * (Complex a, Complex b) {
30     /*
31     # YOUR CODE HERE #
32     */
33     return Complex {a.real * b.real - a.imag * b.imag, a.imag * b.real + a.real * b.imag};
34 }
35
36 Complex operator / (Complex a, Complex b) {
37     /*
38     # YOUR CODE HERE #
39     */
40     double temp = b.real * b.real + b.imag * b.imag;
41     return Complex {(a.real * b.real + a.imag * b.imag) / temp, (a.imag * b.real - a.real * b.imag) / temp};
42 }
43
44 ostream& operator << (ostream& out, const Complex &a) {
45     out << '(' << std::setprecision(2) << a.real << (a.imag >= 0 ? '+' : '-')
46     << std::setprecision(2) << fabs(a.imag) << 'i' << ')';
47     return out;
48 }
49
50 int main() {
51     double real_a, real_b, img_a, img_b;
52     cin >> real_a >> img_a;
53     cin >> real_b >> img_b;
54
55     Complex a(real_a, img_a);
56     Complex b(real_b, img_b);
57
58     cout << a << " + " << b << " = " << a + b << endl;
59     cout << a << " - " << b << " = " << a - b << endl;
60     cout << a << " * " << b << " = " << a * b << endl;
61     cout << a << " / " << b << " = " << a / b << endl;
62
63     return 0;
64 }
65 // Nguyen Van Duy - 20215334
66

```

Chăm thử

Chăm điểm

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình	
✓	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	$(5.5+2i) + (3-1.5i) = (8.5+0.5i)$ $(5.5+2i) - (3-1.5i) = (2.5+3.5i)$ $(5.5+2i) * (3-1.5i) = (20-2.2i)$ $(5.5+2i) / (3-1.5i) = (1.2+1.3i)$	$(5.5+2i) + (3-1.5i) = (8.5+0.5i)$ $(5.5+2i) - (3-1.5i) = (2.5+3.5i)$ $(5.5+2i) * (3-1.5i) = (20-2.2i)$ $(5.5+2i) / (3-1.5i) = (1.2+1.3i)$	✓

Hoàn thành được tất cả các bộ mẫu ✓

Trang tiếp

// Nguyen Van Duy - 20215334

/\*

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

\*/

```
#include <iostream>
```

```
#include <ostream>
```

```
#include <math.h>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
struct Complex {
```

```
    double real;
```

```
    double imag;
```

```
};
```

```
Complex operator + (Complex a, Complex b) {
```

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

```
    # YOUR CODE HERE #
```

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

```
return Complex {a.real + b.real, a.imag + b.imag};  
}
```

Complex operator - (Complex a, Complex b) {

```
    /*****  
    # YOUR CODE HERE #  
    *****/  
    return Complex {a.real - b.real, a.imag - b.imag};  
}
```

Complex operator \* (Complex a, Complex b) {

```
    /*****  
    # YOUR CODE HERE #  
    *****/  
    return Complex {a.real * b.real - a.imag * b.imag, a.imag * b.real + a.real * b.imag};  
}
```

Complex operator / (Complex a, Complex b) {

```
    /*****  
    # YOUR CODE HERE #  
    *****/  
    double temp = b.real * b.real + b.imag * b.imag;  
    return Complex {(a.real * b.real + a.imag * b.imag) / temp, (a.imag * b.real - a.real *  
b.imag) / temp};  
}
```

ostream& operator << (ostream& out, const Complex &a) {

```
    out << '(' << std::setprecision(2) << a.real << (a.imag >= 0 ? '+' : '-')
```

```
<< std::setprecision(2) << fabs(a.imag) << 'i' << ');  
  
return out;  
  
}
```

```
int main() {  
    double real_a, real_b, img_a, img_b;  
    cin >> real_a >> img_a;  
    cin >> real_b >> img_b;  
  
    Complex a{real_a, img_a};  
    Complex b{real_b, img_b};  
  
    cout << a << " + " << b << " = " << a + b << endl;  
    cout << a << " - " << b << " = " << a - b << endl;  
    cout << a << " * " << b << " = " << a * b << endl;  
    cout << a << " / " << b << " = " << a / b << endl;  
  
    return 0;  
}  
  
// Nguyen Van Duy - 20215334
```



**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.

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

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

For example:

Dữ liệu đầu vào	Kết quả
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

**Đáp án:** (penalty regime: 10, 20, ... %)

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.6. Giả thuyết Collatz: bắt đầu từ số dương n bất kỳ,
4 nếu n chẵn thì chia 2, nếu n lẻ thì nhân 3 cộng 1.
5 Hãy viết chương trình mô phỏng lại quá trình biến đổi để
6 kiểm chứng giả thuyết với giá trị của n nhập từ bàn phím.
7 */
8 #include <stdio.h>
9
10
11 void print(int n) {
12     printf("n=%d\n", n);
13 }
14
15 int mul3plus1(int n) {
16     return n * 3 + 1;
17 }
18
19 int div2(int n) {
20     return n / 2;
21 }
22
23 // khai báo các tham số cho các con trỏ hàm odd, even và output
24 void simulate(int n, int (*odd)(int), int (*even)(int), void (*output)(int n))
25 {
26     (*output)(n);
27     if (n == 1) return;
28     if (n % 2 == 0) {
29         n = (*even)(n);
30     } else {
31         n = (*odd)(n);
32     }
33     simulate(n, odd, even, output);
34 }
35
36 int main() {
37     int (*odd)(int) = mul3plus1;
38     int (*even)(int) = div2;

```

## Nguyễn Văn Duy – 20215334

```

39 | /*****
40 |  # YOUR CODE HERE #
41 |  *****/
42 |  int n;
43 |  scanf("%d", &n);
44 |  simulate(n, odd, even, print);
45 |  return 0;
46 | }
47 | // Nguyen Van Duy - 20215334
48 |

```

Chấm thử

Chấm điểm

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình	
✓	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	✓

Hoàn thành được tất cả các bộ mẫu ✓

Trang tiếp

// Nguyen Van Duy - 20215334

/\*

Bài 2.6. Giả thuyết Collatz: bắt đầu từ số dương  $n$  bất kỳ, nếu  $n$  chẵn thì chia 2, nếu  $n$  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 nhập từ bàn phím.

\*/

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

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

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

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

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

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

```
    } else {  
        n = (*odd)(n);  
    }  
    simulate(n, odd, even, output);  
}
```

```
int main() {  
    int (*odd)(int) = mul3plus1;  
    int (*even)(int) = div2;  
    /*****  
    # YOUR CODE HERE #  
    *****/  
    int n;  
    scanf("%d", &n);  
    simulate(n, odd, even, print);  
    return 0;  
}
```

```
// Nguyen Van Duy - 20215334
```

## 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.

**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.

For example:

Dữ liệu đầu vào	Kết quả
5	31 31.4

Đáp án: (penalty regime: 10, 20, ... %)

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.7. Viết hàm tính tổng các phần tử trong hai mảng.
4 Yêu cầu sử dụng function template để cho phép hàm làm
5 việc với các mảng số nguyên lẫn số thực.
6 */
7 #include <iostream>
8
9 using namespace std;
10
11 //# viết hàm arr_sum
12 /*
13 # YOUR CODE HERE #
14 */
15 template <typename T>
16 T arr_sum(T * a, int m, T * b, int n) {
17     T sum = 0;
18
19     // sum a
20     for (int i = 0; i < m; ++i) {
21         sum += a[i];
22     }
23
24     // sum b
25     for (int i = 0; i < n; ++i) {
26         sum += b[i];
27     }
28
29     return sum;
30 }
31
32 int main() {
33     int val;
34     cin >> val;
35
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         double a[] = {3.0, 2, 0, val * 1.0};
43         double b[] = {5, 6.1, 1, 2.3, 7};
44         cout << arr_sum(a, 4, b, 5) << endl;
45     }
46     return 0;
47 }
48 // Nguyen Van Duy - 20215334
49

```

Chấm thử

Chấm điểm

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình	
✓	5	31 31.4	31 31.4	✓
✓	17	43 43.4	43 43.4	✓

```
// Nguyen Van Duy - 20215334
```

```
/*
```

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.

```
*/
```

```
#include <iostream>
```

```
using namespace std;
```

```
//# viết hàm arr_sum
```

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

```
# YOUR CODE HERE #
```

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

```
template <typename T>
```

```
T arr_sum(T * a, int m, T * b, int n) {
```

```
    T sum = 0;
```

```
    // sum a
```

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

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

```
    }
```

```
    // sum b
```

```
for (int i = 0; i < n; ++i) {
    sum += b[i];
}

return sum;
}

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

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

// Nguyen Van Duy - 20215334
```

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

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

For example:

Dữ liệu đầu vào	Kết quả
-10 -5	9 8 15 1 3 7 10 -5 2 3 4 -10

Đáp án: (penalty regime: 10, 20, ... %)

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.8. Viết hàm so sánh cho thuật toán sắp xếp
4 */
5 #include <iostream>
6 #include <vector>
7 #include <algorithm>
8 #include <numeric>
9 using namespace std;
10 int main() {
11     int val1, val2;
12     cin >> val1 >> val2;
13     vector<vector<int>> > a = {
14         {1, 3, 7},
15         {2, 3, 4, val1},
16         {9, 8, 15},
17         {10, val2},
18     };
19     // # sắp xếp các vector trong a theo tổng các phần tử giảm dần
20     // *****
21     # YOUR CODE HERE #
22     // *****
23
24     sort(a.begin(), a.end(), [](vector<int> a, vector<int> b) -> bool {
25         return accumulate(a.begin(), a.end(), 0) > accumulate(b.begin(), b.end(), 0);
26     });
27
28     for (const auto &v : a) {
29         for (int it : v) {
30             cout << it << ' ';
31         }
32         cout << endl;
33     }
34     return 0;
35 // Nguyen Van Duy - 20215334
36

```

Chấm thử

Chấm điểm

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình	
✓	-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	✓

Hoàn thành được tất cả các bộ mẫu ✓

Trang tiếp



// Nguyen Van Duy - 20215334

/\*

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

\*/

#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

using namespace std;

int main() {

int val1, val2;

cin >> val1 >> val2;

vector< vector<int> > 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

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

# YOUR CODE HERE #

\*\*\*\*\*/

sort(a.begin(), a.end(), [](vector<int> a, vector<int> b) -> bool {

return accumulate(a.begin(), a.end(), 0) > accumulate(b.begin(), b.end(), 0);

});

for (const auto &v : a) {

```
    for (int it : v) {  
        cout << it << ' '  
    }  
    cout << endl;  
}  
return 0;  
}  
// Nguyen Van Duy - 20215334
```

## Bài tập 2.9. Viết chương trình in ra tất cả các dãy con của một dãy cho trước.

### Bài 2.9. Tính hàm sigmoid

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

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

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

For example:

Dữ liệu đầu vào	Kết quả
1.5	0.82 Correct answer! Your code is faster at least 30%!

Đáp án: (penalty regime: 10, 20, ... %)

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.9. Dưới đây cung cấp đoạn code đơn giản để tính hàm sigmoid theo công thức trực tiếp.
4 Hãy viết hàm tính xấp xỉ sigmoid(x) đến độ chính xác 10-6 và có tốc độ nhanh hơn ít nhất
5 30% so với code đơn giản.
6 */
7 #include <vector>
8 #include <algorithm>
9 #include <cmath>
10 #include <ctime>
11 #include <algorithm>
12 #include <cstdio>
13
14 using namespace std;
15
16 const int LIMIT = 100;
17 const int NUM_ITER = 100000;
18 const int NUM_INPUTS = NUM_ITER * 100;
19
20 double sigmoid_slow(double x) {
21     return 1.0 / (1.0 + exp(-x));
22 }

```

```

23
24 double x[NUM_INPUTS];
25
26 void prepare_input() {
27     const int PRECISION = 1000000;
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 //# BEGIN fast code
35
36 //# khai báo các biến phụ trợ cần thiết
37 /*****
38 # YOUR CODE HERE #
39 *****/
40 const int TABLE_SIZE = 40001;
41 const double TABLE_MIN = -20;
42 const double TABLE_MAX = 20;
43 const double TABLE_STEP = 0.001;
44 double sigmoid_table[TABLE_SIZE];
45
46 //# hàm chuẩn bị dữ liệu
47 void precalc() {
48     /*****
49     # YOUR CODE HERE #
50     *****/
51     for (int i = 0; i < TABLE_SIZE; i++) {
52         double x = TABLE_MIN + i * TABLE_STEP;
53         sigmoid_table[i] = 1.0 / (1.0 + exp(-x));
54     }
55 }
56
57 //# hàm tính sigmoid(x) nhanh sigmoid_fast(x)

```

```

58 inline double sigmoid_fast(double x) {
59     /*****
60     # YOUR CODE HERE #
61     *****/
62     if (x < TABLE_MIN) return sigmoid_table[0];
63     else if (x > TABLE_MAX) return sigmoid_table[TABLE_SIZE-1];
64     else {
65         double i = (x - TABLE_MIN) / TABLE_STEP;
66         int index = (int) i;
67
68         return sigmoid_table[index] + (sigmoid_table[index+1] - sigmoid_table[index]) * (i - index);
69     }
70 }
71
72 //# END fast code
73
74 double benchmark(double (*calc)(double), vector<double> &result) {
75     const int NUM_TEST = 20;
76
77     double taken = 0;
78     result = vector<double>();
79     result.reserve(NUM_ITER);
80
81     int input_id = 0;
82     clock_t start = clock();
83     for (int t = 0; t < NUM_TEST; ++t) {
84         double sum = 0;
85         for (int i = 0; i < NUM_ITER; ++i) {
86             double v = fabs(calc(x[input_id]));
87             sum += v;
88             if (t == 0) result.push_back(v);
89             if ((++input_id) == NUM_INPUTS) input_id = 0;
90         }
91     }
92     clock_t finish = clock();

```

```

93     taken = (double)(finish - start);
94     //# printf("Time: %.9f\n", taken / CLOCKS_PER_SEC);
95     return taken;
96 }
97
98 bool is_correct(const vector<double> &a, const vector<double> &b) {
99     const double EPS = 1e-6;
100
101     if (a.size() != b.size()) return false;
102     for (int i = 0; i < (int) a.size(); ++i) {
103         if (fabs(a[i] - b[i]) > EPS) {
104             return false;
105         }
106     }
107     return true;
108 }
109
110 int main() {
111     prepare_input();
112     precalc();
113
114     vector<double> a, b;

```

```

115     double slow = benchmark(sigmoid_slow, a);
116     double fast = benchmark(sigmoid_fast, b);
117
118     double xval;
119     scanf("%lf", &xval);
120     printf("%.2f \n", sigmoid_fast(xval));
121
122     if (is_correct(a, b) && (slow/fast > 1.3)) {
123         printf("Correct answer! Your code is faster at least 30%%!\n");
124     } else {
125         printf("Wrong answer or your code is not fast enough!\n");
126     }
127
128     return 0;
129 }
130 // Nguyen Van Duy - 20215334
131

```

Chấm điểm

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình	
✓	1.5	0.82 Correct answer! Your code is faster at least 30%!	0.82 Correct answer! Your code is faster at least 30%!	✓

Kết quả bài lập trình của sinh viên không đúng một hoặc nhiều bộ mẫu ẩn.

Kết thúc bài thi ...

// Nguyen Van Duy - 20215334

/\*

Bài 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  $10^{-6}$  và có tốc độ nhanh hơn ít nhất

30% so với code đơn giản.

\*/

#include <vector>

#include <algorithm>

#include <cmath>

#include <ctime>

#include <algorithm>

#include <cstdio>

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
/*****

# YOUR CODE HERE #

*****/

const int TABLE_SIZE = 40001;
const double TABLE_MIN = -20;
const double TABLE_MAX = 20;
const double TABLE_STEP = 0.001;
double sigmoid_table[TABLE_SIZE];

//# hàm chuẩn bị dữ liệu

void precalc() {

```

```

/*****

# YOUR CODE HERE #

*****/

for (int i = 0; i < TABLE_SIZE; i++) {
    double x = TABLE_MIN + i * TABLE_STEP;
    sigmoid_table[i] = 1.0 / (1.0 + exp(-x));
}

}

//# hàm tính sigmoid(x) nhanh sigmoid_fast(x)
inline double sigmoid_fast(double x) {
    /*****

    # YOUR CODE HERE #

    *****/

    if (x < TABLE_MIN) return sigmoid_table[0];
    else if (x > TABLE_MAX) return sigmoid_table[TABLE_SIZE-1];
    else {
        double i = (x - TABLE_MIN) / TABLE_STEP;
        int index = (int) i;

        return sigmoid_table[index] + (sigmoid_table[index+1] - sigmoid_table[index]) * (i
- index);
    }
}

//# 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 (int i = 0; i < (int) 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;
}
```

}

// Nguyen Van Duy - 20215334

**Bài tập 2.10.** Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ  $N \times N$  theo công thức trực tiếp.

Hãy viết hàm tính tích hai ma trận nhưng có tốc độ nhanh hơn ít nhất 10% so với code đơn giản.

Gợi ý: hãy để ý đến thứ tự truy cập các phần tử trong ma trận, tối ưu cache hoặc sử dụng thuật toán tốt hơn  $O(N^3)$ .

```

CMakeLists.txt  main.cpp x
1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.10. Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ  $N \times N$  theo công thức trực tiếp.
4 Hãy viết hàm tính tích hai ma trận nhưng có tốc độ nhanh hơn ít nhất 10% so với code đơn giản.
5 Gợi ý: hãy để ý đến thứ tự truy cập các phần tử trong ma trận, tối ưu cache hoặc sử dụng thuật toán tốt hơn  $O(N^3)$ 
6 */
7 #include <iostream>
8 #include <cstring>
9 using namespace std;
10 const int N = 128;
11 struct Matrix {
12     unsigned int mat[N][N];
13     Matrix() {
14         memset(&mat, 0, sizeof mat);
15     }
16 };
17 bool operator == (const Matrix &a, const Matrix &b) {
18     for (int i = 0; i < N; ++i) {
19         for (int j = 0; j < N; ++j) {
20             if (a.mat[i][j] != b.mat[i][j]) return false;
21         }
22     }
23     return true;
24 }
25 Matrix multiply_naive(const Matrix &a, const Matrix &b) {
26     Matrix c;
27     for (int i = 0; i < N; ++i) {
28         for (int j = 0; j < N; ++j) {
29             for (int k = 0; k < N; ++k) {
30                 c.mat[i][j] += a.mat[i][k] * b.mat[k][j];
31             }
32         }
33     }
34     return c;
35 }
36 Matrix multiply_fast(const Matrix &a, const Matrix &b) {
37     /******
38     # YOUR CODE HERE #
39     *****/
40     Matrix c; // c = a x b
41     for (int i = 0; i < N; ++i) {
42         for (int j = 0; j < i+1; ++j) {
43             c.mat[i][j] += a.mat[i][j] * b.mat[j][i]; // the main diagonal
44         }
45         for (int j = i+1; j < N; ++j) {
46             c.mat[i][j] += a.mat[i][j] * b.mat[j][i]; // the main diagonal
47             for (int k = 0; k < N; ++k) {
48                 c.mat[i][j] += a.mat[i][k] * b.mat[k][j]; // calc c_ij
49                 c.mat[j][i] += a.mat[j][k] * b.mat[k][i]; // calc c_ji
50             }
51         }
52     }
53     return c;
54 }

```

```

55 Matrix gen_random_matrix() {
56     Matrix a;
57     for (int i = 0; i < N; ++i) {
58         for (int j = 0; j < N; ++j) {
59             a.mat[i][j] = rand();
60         }
61     }
62     return a;
63 }
64 Matrix base;
65 double benchmark(Matrix (*multiply) (const Matrix&, const Matrix&), Matrix &result) {
66     const int NUM_TEST = 10;
67     const int NUM_ITER = 64;
68     Matrix a = base;
69     result = a;
70     double taken = 0;
71     for (int t = 0; t < NUM_TEST; ++t) {
72         clock_t start = clock();
73         for (int i = 0; i < NUM_ITER; ++i) {
74             a = multiply(a, result);
75             result = multiply(result, a);
76         }
77         clock_t finish = clock();
78         taken += (double)(finish - start);
79     }
80     taken /= NUM_TEST;
81     printf( format: "Time: %.9f\n", taken / CLOCKS_PER_SEC);
82     return taken;
83 }
84 int main() {
85     base = gen_random_matrix();
86     Matrix a, b;
87     printf( format: "Slow version\n");
88     double slow = benchmark( multiply: multiply_naive, &a);
89     printf( format: "Fast version\n");
90     double fast = benchmark( multiply: multiply_fast, &b);
91     if (a == b) {
92         printf( format: "Correct answer! Your code is %.2f%% faster\n", slow / fast * 100.0);
93     } else {
94         printf( format: "Wrong answer!\n");
95     }
96     return 0;
97 }
98 // Nguyen Van Duy - 20215334
99

```

Run test x

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

Slow version

Time: 1.252800000

Fast version

Time: 1.007600000

Correct answer! Your code is 124.34% faster

Process finished with exit code 0

test

// Nguyen Van Duy - 20215334

/\*

Bài 2.10. Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ NxN theo công thức trực tiếp.

Hãy viết hàm tính tích hai ma trận nhưng có tốc độ nhanh hơn ít nhất 10% so với code đơn giản.

Gợi ý: hãy để ý đến thứ tự truy cập các phần tử trong ma trận, tối ưu cache hoặc sử dụng thuật toán tốt hơn  $O(N^3)$

```
*/

#include <iostream>

#include <cstring>

using namespace std;

const int N = 128;

struct Matrix {

    unsigned int mat[N][N];

    Matrix() {

        memset(mat, 0, sizeof mat);

    }

};

bool operator == (const Matrix &a, const Matrix &b) {

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

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

            if (a.mat[i][j] != b.mat[i][j]) return false;

        }

    }

    return true;

}
```

```
}
```

```
Matrix multiply_naive(const Matrix &a, const Matrix &b) {
```

```
    Matrix c;
```

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

```
        for (int j = 0; j < N; ++j) {
```

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

```
                c.mat[i][j] += a.mat[i][k] * b.mat[k][j];
```

```
            }
```

```
        }
```

```
    }
```

```
    return c;
```

```
}
```

```
Matrix multiply_fast(const Matrix &a, const Matrix &b) {
```

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

```
    # YOUR CODE HERE #
```

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

```
    Matrix c; // c = a x b
```

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

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

```
            c.mat[i][i] += a.mat[i][j] * b.mat[j][i]; // the main diagonal
```

```
        }
```

```
        for (int j = i+1; j < N; ++j) {
```

```
            c.mat[i][i] += a.mat[i][j] * b.mat[j][i]; // the main diagonal
```

```
for (int k = 0; k < N; ++k) {  
  
    c.mat[i][j] += a.mat[i][k] * b.mat[k][j]; // calc c_ij  
  
    c.mat[j][i] += a.mat[j][k] * b.mat[k][i]; // calc c_ji  
  
}  
  
}  
  
}  
  
return c;  
  
}  
  
Matrix gen_random_matrix() {  
  
    Matrix a;  
  
    for (int i = 0; i < N; ++i) {  
  
        for (int j = 0; j < N; ++j) {  
  
            a.mat[i][j] = rand();  
  
        }  
  
    }  
  
    return a;  
  
}  
  
Matrix base;  
  
double benchmark(Matrix (*multiply) (const Matrix&, const Matrix&), Matrix &result) {  
  
    const int NUM_TEST = 10;  
  
    const int NUM_ITER = 64;  
  
    Matrix a = base;  
  
    result = a;
```

```
double taken = 0;

for (int t = 0; t < NUM_TEST; ++t) {

    clock_t start = clock();

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

        a = multiply(a, result);

        result = multiply(result, a);

    }

    clock_t finish = clock();

    taken += (double)(finish - start);

}

taken /= NUM_TEST;

printf("Time: %.9f\n", taken / CLOCKS_PER_SEC);

return taken;

}

int main() {

    base = gen_random_matrix();

    Matrix a, b;

    printf("Slow version\n");

    double slow = benchmark(multiply_naive, a);

    printf("Fast version\n");

    double fast = benchmark(multiply_fast, b);

    if (a == b) {

        printf("Correct answer! Your code is %.2f%% faster\n", slow / fast * 100.0);
```



```
} else {  
  
    printf("Wrong answer!\n");  
  
}  
  
return 0;  
  
}  
  
// Nguyen Van Duy – 20215334
```

**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 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.11. Cho 2 đa thức A(x) và B(x) tương ứng có bậc N và M.
4 Hãy tính ma trận tích C(x) = A(x) * B(x) có bậc N + M.
5 */
6 #include <iostream>
7 #include <vector>
8
9 using namespace std;
10
11 class Polynomial {
12 private:
13     int n;
14     vector<int> a;
15
16 public:
17     // no-args constructor
18     Polynomial() {
19         this->n = 0;
20         a.push_back(0);
21     }
22
23     // Parameterized constructor
24     explicit Polynomial(int n) : n(n) {
25         for (int i = 0; i <= n; ++i) {
26             a.push_back(0);
27         }
28     }
29
30     // Parameterized constructor
31     Polynomial(int n, const vector<int> &a) : n(n), a(a) {}
32
33     // return size of vector = N + 1
34     [[nodiscard]] size_t size() const {
35         return a.size();
36     }
37
38     // XOR
39     [[nodiscard]] int xor_coefficients() const {
40         int res = 0;
41
42         for (int i = 0; i < a.size(); ++i) {
43             res ^= a[i];
44         }
45
46         return res;
47     }
48
49     [[nodiscard]] int getN() const {
50         return n;
51     }
52
53     void clear() {
54         n = 0;
55         a.clear();
56     }
57
58     friend istream & operator >> (istream & is, Polynomial & v) {
59         v.clear();
60         is >> v.n;
61
62         for (int i = 0; i <= v.n; ++i) {
63             int temp;
64             is >> temp;
65             v.a.push_back(temp);
66         }
67
68         return is;
69     }

```

// Nguyen Van Duy - 20215334

/\*

Bài 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  $N + M$ .

```
*/
```

```
#include <iostream>
```

```
#include <vector>
```

```
using namespace std;
```

```
class Polynomial {
```

```
private:
```

```
    int n;
```

```
    vector<int> a;
```

```
public:
```

```
    // no-args constructor
```

```
    Polynomial() {
```

```
        this->n = 0;
```

```
        a.push_back(0);
```

```
    }
```

```
    // Parameterized constructor
```

```
    explicit Polynomial(int n) : n(n) {
```

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

```
            a.push_back(0);
```

```
    }  
}  
  
// Parameterized constructor  
Polynomial(int n, const vector<int> &a) : n(n), a(a) {}  
  
// return size of vector = N + 1  
[[nodiscard]] size_t size() const {  
    return a.size();  
}  
  
// XOR  
[[nodiscard]] int xor_coefficients() const {  
    int res = 0;  
  
    for (int i = 0; i < a.size(); ++i) {  
        res ^= a[i];  
    }  
  
    return res;  
}  
  
[[nodiscard]] int getN() const {
```

```
    return n;
}

void clear() {
    n = 0;
    a.clear();
}

friend istream & operator >> (istream & is, Polynomial & v) {
    v.clear();
    is >> v.n;

    for (int i = 0; i <= v.n; ++i) {
        int temp;
        is >> temp;
        v.a.push_back(temp);
    }

    return is;
}

friend ostream & operator << (ostream & os, Polynomial v) {
    for (int i = 0; i < v.n; ++i) {
```

```
        os << v.a[i] << " ";

    }

    os << v.a[v.n];

    return os;

}

// multiplying two polynomials

Polynomial operator * (const Polynomial & v) const {

    Polynomial res(this->n + v.n);

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

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

            if (j < this->size() && i-j < v.size()) {

                res.a[res.getN() - i] += this->a[this->getN()-j] * v.a[v.getN()-(i-j)];

            }

        }

    }

    return res;

};
```

```
int main() {  
    Polynomial a, b;  
  
    // enter 2 polynomials  
    cin >> a >> b;  
  
    // XOR of the coefficients of the polynomial  
    cout << (a*b).xor_coefficients() << endl;  
  
    return 0;  
}  
  
// Nguyen Van Duy - 20215334
```



**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.

```
CL  test  Version control  Debug  test
CMakeLists.txt  main.cpp
1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 2.12. Hôm nay, cô giáo giao cho An một câu hỏi học búa.
4 Cô cho một danh sách với mỗi phần tử có dạng <key, value> và
5 yêu cầu An sắp xếp danh sách đó giảm dần theo giá trị value.
6 Nếu 2 phần tử có value giống nhau thì sắp xếp giảm dần theo key.
7
8 Hãy viết một chương trình sử dụng hàm nặc danh để giúp An làm bài tập.
9 */
10 #include <iostream>
11 #include <sstream>
12 #include <algorithm>
13 #include <vector>
14 #include <map>
15
16 using namespace std;
17
18 int main() {
19     vector<pair<int, int>> list;
20     map<int, int> index;
21
22     // input : key and value in line
23     for (int i = 0; true; ++i) {
24         static string s;
25         getline(&cin, &s);
26
27         if (s.empty()) {
28             break;
29         } else {
30             int key, value;
31             stringstream ss(s);
32
33             // get key and value
34             ss >> key >> value;
35
36             if (index.find(key) == index.end()) {
37                 // key does not exist
38                 list.emplace_back(key, value);
39                 index[key] = i;
40             } else {
41                 // key exists
42                 // update : change value
43                 list[index[key]] = make_pair(key, value);
44                 --i;
45             }
46         }
47     }
48
49     // sort by value
50     sort(list.begin(), list.end(), [](pair<int, int> a, pair<int, int> b) -> bool {
51         if (a.second > b.second) return true;
52         if (a.second < b.second) return false;
53
54         // 2 values are equal
55         // sort by key
56         return a.first > b.first;
57     });
58 }
```

```
59
60 // display list
61 for (auto i : pair<int, int> : list) {
62     cout << i.first << " " << i.second << endl;
63 }
64
65 return 0;
66 }
67 // Nguyen Van Duy - 20215334
68
```

Run test

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

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

Process finished with exit code 0

test

```
// Nguyen Van Duy - 20215334
```

```
/*
```

Bài 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.

```
*/
```

```
#include <iostream>
```

```
#include <sstream>
```

```
#include <algorithm>
```

```
#include <vector>
```

```
#include <map>
```

```
using namespace std;
```

```
int main() {
```

```
    vector<pair<int, int>> list;
```

```
    map<int, int> index;
```

```
    // input : key and value in line
```

```
    for (int i = 0; true; ++i) {
```

```
        static string s;
```

```
        getline(cin, s);
```

```
        if (s.empty()) {
```

```
            break;
```

```
        } else {
```

```
            int key, value;
```

```
            stringstream ss(s);
```

```
            // get key and value
```

```
ss >> key >> value;
```

```
if (index.find(key) == index.end()) {  
    // key does not exist  
    list.emplace_back(key, value);  
    index[key] = i;  
} else {  
    // key exists  
    // update : change value  
    list[index[key]] = make_pair(key, value);  
    --i;  
}  
  
}  
  
}
```

```
// sort by value  
sort(list.begin(), list.end(), [](pair<int, int> a, pair<int, int> b) -> bool {  
    if (a.second > b.second) return true;  
    if (a.second < b.second) return false;  
  
    // 2 values are equal  
  
    // sort by key
```

```
    return a.first > b.first;

});

// display list
for (auto i : list) {
    cout << i.first << " " << i.second << endl;
}

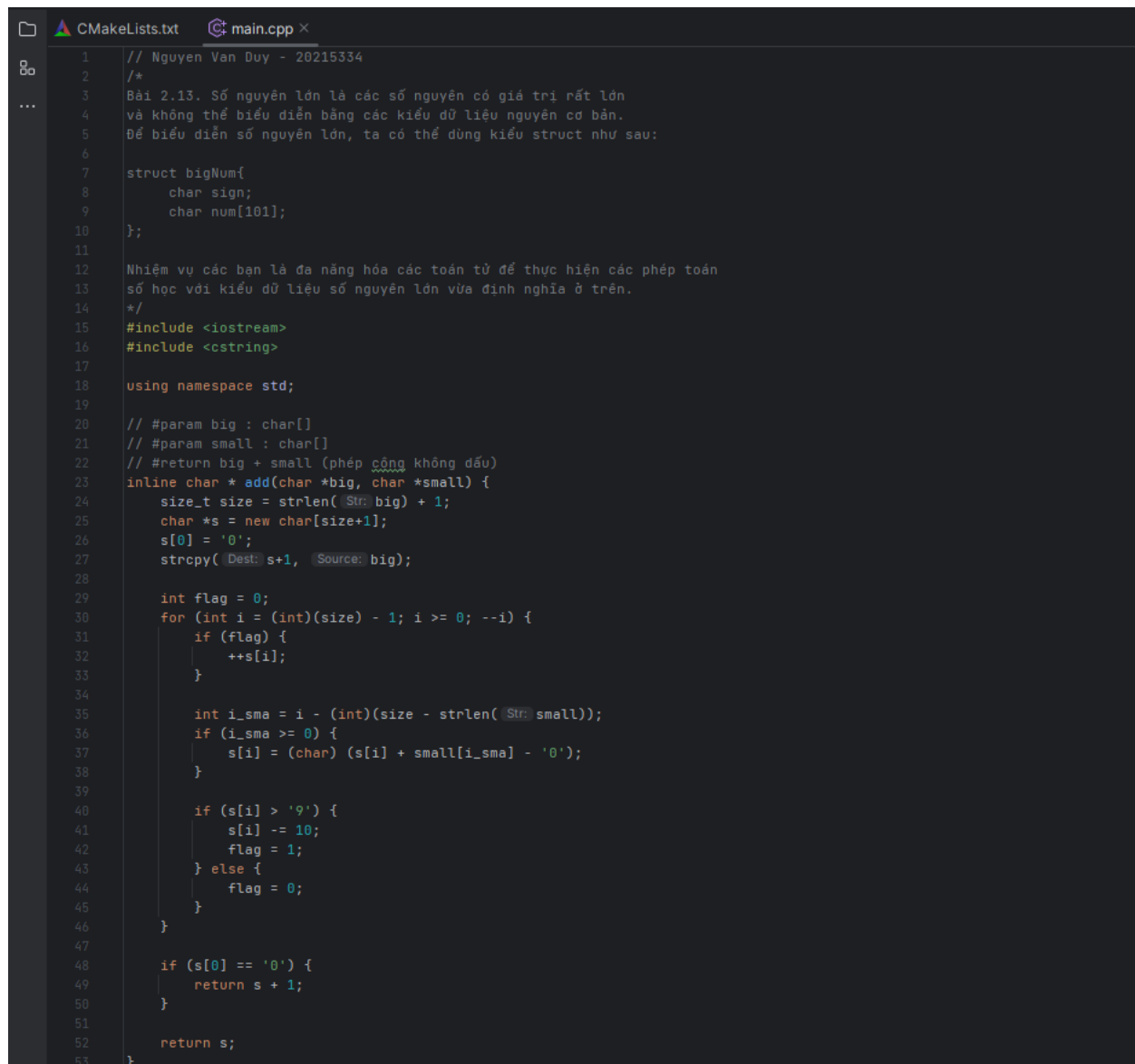
return 0;
}

// Nguyen Van Duy - 20215334
```

**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 // Nguyen Van Duy - 20215334  
2 /*  
3 Bài 2.13. Số nguyên lớn là các số nguyên có giá trị rất lớn  
4 và không thể biểu diễn bằng các kiểu dữ liệu nguyên cơ bản.  
5 Để biểu diễn số nguyên lớn, ta có thể dùng kiểu struct như sau:  
6  
7 struct bigNum{  
8     char sign;  
9     char num[101];  
10 };  
11  
12 Nhiệm vụ các bạn là đa năng hóa các toán tử để thực hiện các phép toán  
13 số học với kiểu dữ liệu số nguyên lớn vừa định nghĩa ở trên.  
14 */  
15 #include <iostream>  
16 #include <cstring>  
17  
18 using namespace std;  
19  
20 // #param big : char[]  
21 // #param small : char[]  
22 // #return big + small (phép cộng không dấu)  
23 inline char * add(char *big, char *small) {  
24     size_t size = strlen( Str: big) + 1;  
25     char *s = new char[size+1];  
26     s[0] = '0';  
27     strcpy( Dest: s+1, Source: big);  
28  
29     int flag = 0;  
30     for (int i = (int)(size) - 1; i >= 0; --i) {  
31         if (flag) {  
32             ++s[i];  
33         }  
34  
35         int i_sma = i - (int)(size - strlen( Str: small));  
36         if (i_sma >= 0) {  
37             s[i] = (char) (s[i] + small[i_sma] - '0');  
38         }  
39  
40         if (s[i] > '9') {  
41             s[i] -= 10;  
42             flag = 1;  
43         } else {  
44             flag = 0;  
45         }  
46     }  
47  
48     if (s[0] == '0') {  
49         return s + 1;  
50     }  
51  
52     return s;  
53 }
```

```

54
55 // #param big : char[] --> số lớn hơn
56 // #param small : char[] --> số nhỏ hơn
57 // #return big - small (phép trừ không dấu)
58 inline char * sub(char *big, char *small) {
59     size_t size = strlen( Str: big);
60     char *s = new char[size+1];
61     strcpy( Dest: s, Source: big);
62
63     int flag = 0;
64     for (int i = (int)(size) - 1; i >= 0; --i) {
65         if (flag) {
66             --s[i];
67         }
68
69         int i_sma = i - (int)(size - strlen( Str: small));
70         if (i_sma >= 0) {
71             s[i] = (char) (s[i] - small[i_sma] + '0');
72         }
73
74         if (s[i] < '0') {
75             s[i] += 10;
76             flag = -1;
77         } else {
78             flag = 0;
79         }
80     }
81
82     return s;
83 }
84
85 struct bigNum{
86     char sign;
87     char num[101];
88
89     // no-args constructor
90     bigNum() {
91         sign = '1';
92         num[0] = '0';
93         num[1] = '\0';
94     }
95
96     // Parameterized constructor
97     bigNum(char sign, const char *num) {
98         this->sign = sign;
99         for (int i = 0; i < 101; ++i) {
100             this->num[i] = num[i];
101         }
102     }
103
104     // Parameterized constructor
105     bigNum(long long v) {
106         *this = v;
107     }
108
109     // absolute value
110     bigNum abs() const {
111         bigNum res = *this;
112         res.sign = '1';
113         return res;
114     }
115
116     // Opposite number
117     bigNum operator - () const {
118         bigNum res = *this;
119         res.sign = (char) (97 - sign);
120         return res;
121     }
122
123     // bigNum = bigNum
124     bigNum & operator = (const bigNum v) {
125         sign = v.sign;
126         memcpy( Dest: num, Src: v.num, Size: 101);
127         return *this;
128     }

```

```

129 // bigNum = long long int
130
131 bigNum & operator = (long long int v) {
132     if (v < 0) {
133         sign = '0';
134         v = -v;
135     } else {
136         sign = '1';
137     }
138
139     char temp[100];
140     int len = 0;
141     v /= 10;
142 }
143
144 this->num[len--] = '\0';
145 for (int i = 0; i <= len; ++i) {
146
147     return *this;
148 }
149
150 friend istream & operator >> (istream & is, bigNum & bNum) {
151     is >> bNum.sign >> bNum.num;
152     return is;
153 }
154
155 friend ostream & operator << (ostream & os, bigNum bNum) {
156     // if (bNum.sign == '0') {
157     //     os << "-";
158     // }
159     // os << bNum.num;
160     os << bNum.sign << bNum.num;
161     return os;
162 }
163
164 // #param a : bigNum
165 // #param b : bigNum
166 // #return a.abs() > b.abs() (so sánh không dấu)
167 static inline bool biggerNum(bigNum a, bigNum b) {
168     size_t len1 = strlen(a.num);
169     size_t len2 = strlen(b.num);
170
171     if (len1 > len2) return true;
172     if (len1 < len2) return false;
173
174     for (int i = 0; i < 101; ++i) {
175         if (a.num[i] > b.num[i]) return true;
176         if (a.num[i] < b.num[i]) return false;
177     }
178
179     return false;
180 }
181
182 bigNum operator + (bigNum other) {
183     bigNum res;
184
185     bigNum *big, *small;
186     if (biggerNum(a: *this, b: other)) {
187         big = this;
188         small = &other;
189     } else {
190         big = &other;
191         small = this;
192     }
193
194     res.sign = big->sign;
195     if (small->sign == big->sign) {
196         strcpy(Dest: res.num, Source: add(big: big->num, small: small->num));
197     } else {
198         strcpy(Dest: res.num, Source: sub(big: big->num, small: small->num));
199     }
200
201     return res;
202 }
203
204 bigNum operator - (bigNum other) {
205     return *this + (-other);
206 }

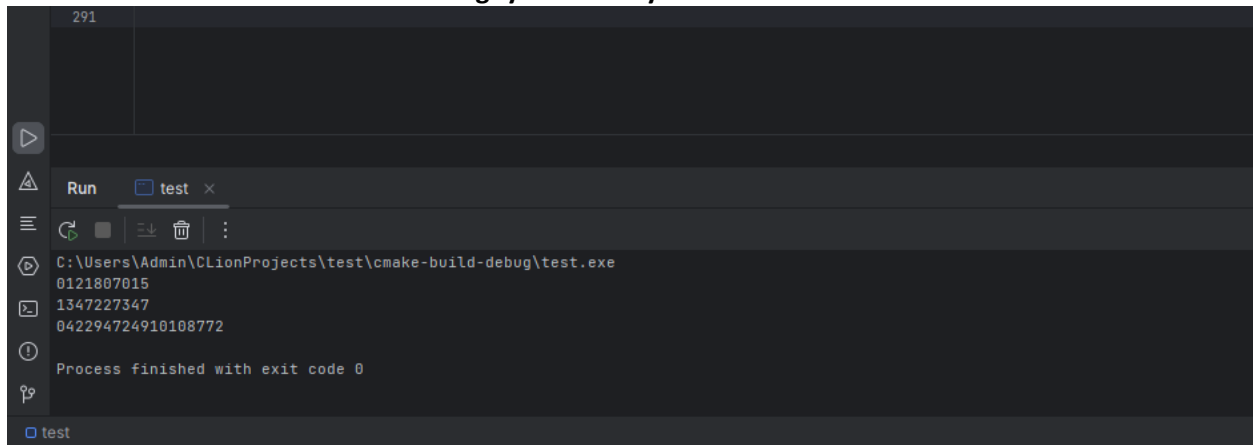
```



```

212
213 // bigNum x bigNum
214 bigNum operator * (bigNum other) {
215     int len_this = (int) strlen( Str: this->num);
216     int len_other = (int) strlen( Str: other.num);
217     int len = len_this + len_other;
218
219     char *temp = new char[len+1];
220     temp[len] = '\0';
221
222     int flag = 0;
223     int last = len - 1;
224     int last_this = len_this - 1;
225     int last_other = len_other - 1;
226     for (int i = 0; i < len; ++i) {
227         int calc = flag % 10;
228         flag /= 10;
229
230         for (int j = 0; j <= i; ++j) {
231             if (j < len_this && i-j < len_other) {
232                 calc += (this->num[last_this-j] - '0') * (other.num[last_other-(i-j)] - '0');
233             }
234         }
235
236         if (calc > 9) {
237             flag += calc / 10;
238             calc %= 10;
239         }
240
241         temp[last - i] = (char) (calc + '0');
242     }
243
244     return {sign: (this->sign == other.sign) ? '1' : '0', num: (*temp - '0') ? temp : temp + 1};
245 }
246
247 // bigNum x other(Integer)
248 template<class T>
249 bigNum operator * (T v) const {
250     char sign_res = sign;
251     if (v < 0) {
252         sign_res = (char) (97 - sign);
253         v = -v;
254     }
255
256     char temp[101];
257     int len = (int) strlen( Str: num);
258     memset( Dst: temp, Val: '0', Size: 101-len-1);
259     memcpy( Dst: temp+101-len-1, Src: num, Size: len+1);
260
261     int flag = 0;
262     int i;
263     for (i = 99; i >= 0; --i) {
264         int calc = (int)(temp[i] - '0') * v + flag % 10;
265         flag /= 10;
266
267         if (calc > 9) {
268             flag += calc / 10;
269             calc %= 10;
270         }
271
272         temp[i] = (char) (calc + '0');
273
274         if (99-i >= len && !calc && !flag) {
275             ++i;
276             break;
277         }
278     }
279
280     return {sign: sign_res, num: temp + i};
281 }
282 };
283
284 int main() {
285     bigNum a, b;
286     cin >> a >> b;
287     cout << a*b - a*3 + b*4 << endl;
288     return 0;
289 }
290 // Nguyen Van Duy - 20215334
291

```



```
// Nguyen Van Duy - 20215334
```

```
/*
```

Bài 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.

```
*/
```

```
#include <iostream>
```

```
#include <cstring>
```

```
using namespace std;
```

```
// #param big : char[]
```

```
// #param small : char[]
```

```
// #return big + small (phép cộng không dấu)
```

```
inline char * add(char *big, char *small) {
```

```
    size_t size = strlen(big) + 1;
```

```
    char *s = new char[size+1];
```

```
    s[0] = '0';
```

```
    strcpy(s+1, big);
```

```
    int flag = 0;
```

```
    for (int i = (int)(size) - 1; i >= 0; --i) {
```

```
        if (flag) {
```

```
            ++s[i];
```

```
        }
```

```
        int i_sma = i - (int)(size - strlen(small));
```

```
        if (i_sma >= 0) {
```

```
            s[i] = (char) (s[i] + small[i_sma] - '0');
```

```
        }
```

```
        if (s[i] > '9') {
```

```
            s[i] -= 10;
```

```
    flag = 1;

    } else {

        flag = 0;

    }

}
```

```
if (s[0] == '0') {

    return s + 1;

}
```

```
return s;

}
```

```
// #param big : char[] --> số lớn hơn
// #param small : char[] --> số nhỏ hơn
// #return big - small (phép trừ không dấu)
inline char * sub(char *big, char *small) {

    size_t size = strlen(big);

    char *s = new char[size+1];

    strcpy(s, big);

    int flag = 0;

    for (int i = (int)(size) - 1; i >= 0; --i) {

        if (flag) {
```

```
--s[i];

}

int i_sma = i - (int)(size - strlen(small));

if (i_sma >= 0) {

    s[i] = (char) (s[i] - small[i_sma] + '0');

}

if (s[i] < '0') {

    s[i] += 10;

    flag = -1;

} else {

    flag = 0;

}

}

return s;

}
```

```
struct bigNum{

    char sign;

    char num[101]{};

    // no-args constructor
```

```
bigNum() {
```

```
    sign = '1';
```

```
    num[0] = '0';
```

```
    num[1] = '\0';
```

```
}
```

```
// Parameterized constructor
```

```
bigNum(char sign, const char *num) {
```

```
    this->sign = sign;
```

```
    for (int i = 0; i < 101; ++i) {
```

```
        this->num[i] = num[i];
```

```
    }
```

```
}
```

```
// Parameterized constructor
```

```
bigNum(long long v) {
```

```
    *this = v;
```

```
}
```

```
// absolute value
```

```
bigNum abs() const {
```

```
    bigNum res = *this;
```

```
    res.sign = '1';
```

```
    return res;
```

```
}
```

```
// Opposite number
```

```
bigNum operator - () const {
```

```
    bigNum res = *this;
```

```
    res.sign = (char) (97 - sign);
```

```
    return res;
```

```
}
```

```
// bigNum = bigNum
```

```
bigNum & operator = (const bigNum v) {
```

```
    sign = v.sign;
```

```
    memcpy(num, v.num, 101);
```

```
    return *this;
```

```
}
```

```
// bigNum = long long int
```

```
bigNum & operator = (long long int v) {
```

```
    if (v < 0) {
```

```
        sign = '0';
```

```
        v = -v;
```

```
    } else {
```

```
        sign = '1';
```

```
    }
```

```
char temp[100];

int len = 0;

for (int i = 0; i < 100 && v; ++i) {

    temp[i] = (char) (v % 10 + '0');

    ++len;

    v /= 10;

}

this->num[len--] = '\0';

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

    num[len-i] = temp[i];

}

return *this;

}

friend istream & operator >> (istream & is, bigNum & bNum) {

    is >> bNum.sign >> bNum.num;

    return is;

}

friend ostream & operator << (ostream & os, bigNum bNum) {

//    if (bNum.sign == '0') {
```



```
//      os << "-";

//      }

//      os << bNum.num;

      os << bNum.sign << bNum.num;

      return os;

}


// #param a : bigNum

// #param b : bigNum

// #return a.abs() > b.abs() (so sánh không dấu)

static inline bool biggerNum(bigNum a, bigNum b) {

    size_t len1 = strlen(a.num);

    size_t len2 = strlen(b.num);

    if (len1 > len2) return true;

    if (len1 < len2) return false;

    for (int i = 0; i < 101; ++i) {

        if (a.num[i] > b.num[i]) return true;

        if (a.num[i] < b.num[i]) return false;

    }

    return false;

}
```

```
bigNum operator + (bigNum other) {  
    bigNum res;  
  
    bigNum *big, *small;  
    if (biggerNum(*this, other)) {  
        big = this;  
        small = &other;  
    } else {  
        big = &other;  
        small = this;  
    }  
  
    res.sign = big->sign;  
    if (small->sign == big->sign) {  
        strcpy(res.num, add(big->num, small->num));  
    } else {  
        strcpy(res.num, sub(big->num, small->num));  
    }  
  
    return res;  
}  
  
bigNum operator - (bigNum other) {
```

```
return *this + (-other);

}

// bigNum x bigNum

bigNum operator * (bigNum other) {

    int len_this = (int) strlen(this->num);

    int len_other = (int) strlen(other.num);

    int len = len_this + len_other;

    char *temp = new char[len+1];

    temp[len] = '\0';

    int flag = 0;

    int last = len - 1;

    int last_this = len_this - 1;

    int last_other = len_other - 1;

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

        int calc = flag % 10;

        flag /= 10;

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

            if (j < len_this && i-j < len_other) {

                calc += (this->num[last_this-j] - '0') * (other.num[last_other-(i-j)] - '0');

            }

        }

    }

}
```

```
}
```

```
if (calc > 9) {
```

```
    flag += calc / 10;
```

```
    calc %= 10;
```

```
}
```

```
temp[last - i] = (char) (calc + '0');
```

```
}
```

```
return {(this->sign == other.sign) ? '1' : '0', (*temp - '0') ? temp : temp + 1};
```

```
}
```

```
// bigNum x other(Integer)
```

```
template<class T>
```

```
bigNum operator * (T v) const {
```

```
    char sign_res = sign;
```

```
    if (v < 0) {
```

```
        sign_res = (char) (97 - sign);
```

```
        v = -v;
```

```
}
```

```
char temp[101];
```

```
int len = (int) strlen(num);
```

```
memset(temp, '0', 101-len-1);

memcpy(temp+101-len-1, num, len+1);


int flag = 0;

int i;

for (i = 99; i >= 0; --i) {

    int calc = (int)(temp[i] - '0') * v + flag % 10;

    flag /= 10;


    if (calc > 9) {

        flag += calc / 10;

        calc %= 10;

    }


    temp[i] = (char) (calc + '0');


    if (99-i >= len && !calc && !flag) {

        ++i;

        break;

    }

}


return {sign_res, temp + i};

}
```

```
};
```

```
int main() {
```

```
    bigNum a, b;
```

```
    cin >> a >> b;
```

```
    cout << a*b - a*3 + b*4 << endl;
```

```
    return 0;
```

```
}
```

```
// Nguyen Van Duy - 20215334
```