Exercise 1

Exercise 1

Find **PRE- and POST-conditions** for this function.

1. Function:

```
double f (double i,
          double j,
          double k)
{
  if (i > j)
    if (i > k)
      return i;
    else
      return k;
  else
    if (j > k)
      return j;
    else
      return k;
}
```

Exercise 1

PRE-Condition:
(not needed)

POST-Condition:

```
// POST: return value is
//       the maximum of
//       i,j and k
```

1. Function:

```
double f (double i,
          double j,
          double k)
{
  if (i > j)
    if (i > k)
      return i;
    else
      return k;
  else
    if (j > k)
      return j;
    else
      return k;
}
```

(From: Script Exercise 79)

Exercise 1

Find **PRE- and POST-conditions** for this function.

2. Function:

```
double g (int i, int j)
{
  double r = 0.0;
  for (int k = i; k <= j; ++k)
    r += 1.0 / k;
  return r;
}
```

Exercise 1

2. Function:

```
double g (int i, int j)
{
  double r = 0.0;
  for (int k = i; k <= j; ++k)
    r += 1.0 / k;
  return r;
}
```

PRE-Condition: `// PRE: 0 not contained in {i, ..., j}`

POST-Condition:
```
// POST: return value is the sum
//          1/i + 1/(i+1) + ... + 1/j
```

Exercise 2

Exercise 2

- Fix the **problems** in the following functions.
- Then add suitable **PRE- and POST-conditions**.

1. Function:

```
bool is_even (int i)
{
    if (i % 2 == 0) return true;
}
```

Exercise 2

- Problem: just a return value for even inputs

1. Function:

```
bool is_even (int i)
{
    if (i % 2 == 0) return true;
}
```

(From: Script Exercise 80)

Exercise 2

- Problem: just a return value for even inputs
- Fix: e.g. direct return of `i % 2 == 0`

1. Function:

```
bool is_even (int i)
{
    if (i % 2 == 0) return true;
}
```

```
bool is_even (int i)
{
    return (i % 2 == 0);
}
```

Exercise 2

- Problem: just a return value for even inputs
- Fix: e.g. direct return of `i % 2 == 0`

1. Function:

```
bool is_even (int i)
{
    if (i % 2 == 0) return true;
}
```

```
bool is_even (int i)
{
    return (i % 2 == 0);
}
```

PRE-Condition: (not needed)

POST-Condition:
```
// POST: return value is true if and only
//       if i is even
```

Exercise 2

- Fix the **problems** in the following functions.
- Then add suitable **PRE- and POST-conditions**.

2. Function:

```
double inverse (double x) {
    double result;
    if (x != 0.0)
         result = 1.0 / x;
    return result;
}
```

(From: Script Exercise 80)

Exercise 2

- Problem: no return value for x=0

2. Function:

```
double inverse (double x) {
    double result;
    if (x != 0.0)
         result = 1.0 / x;
    return result;
}
```

Exercise 2

- Problem: no return value for `x=0`
- Fix: `x != 0.0` as PRE-condition (and `assert`)

2. Function:

```
double inverse (double x) {
    double result;
    if (x != 0.0)
         result = 1.0 / x;
    return result;
}
```

```
// PRE: x != 0.0
// POST: ...
double inverse (double x) {
    assert(x != 0.0);
    return 1.0 / x;
}
```

Exercise 2

- Problem: no return value for `x=0`
- Fix: `x != 0.0` as PRE-condition (and `assert`)

2. Function:

```
double inverse (double x) {
    double result;
    if (x != 0.0)
        result = 1.0 / x;
    return result;
}
```

```
// PRE: x != 0.0
// POST: ...
double inverse (double x) {
    assert(x != 0.0);
    return 1.0 / x;
}
```

PRE-Condition: `// PRE: x != 0.0`

POST-Condition: `// POST: return value is 1/x`

Exercise 2

Another solution:

Exercise 2

Another solution:

`else` with special return value

```
double inverse (double x)
{
    double result;
    if (x != 0.0)
        result = 1.0 / x;
    else
        result = 0.0;
    return result;
}
```

Exercise 2

Another solution:

`else` with special return value

```
double inverse (double x)
{
    double result;
    if (x != 0.0)
        result = 1.0 / x;
    else
        result = 0.0;
    return result;
}
```

PRE-Condition: (not needed)

POST-Condition:
```
// POST: return value is 1/x if x!=0.0
//       return value is 0.0 else
```

(From: Script Exercise 80)

Exercise 3

Exercise 3

- What is the **output** of this program?
- You can neglect possible over- or underflows for this exercise.

```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

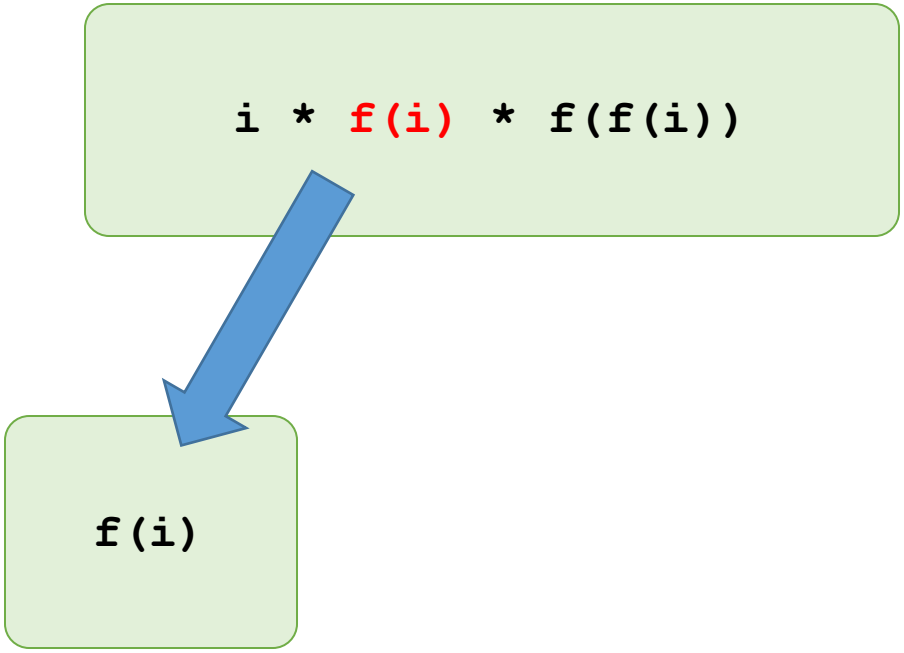
Exercise 3

```
i * f(i) * f(f(i))
```

```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

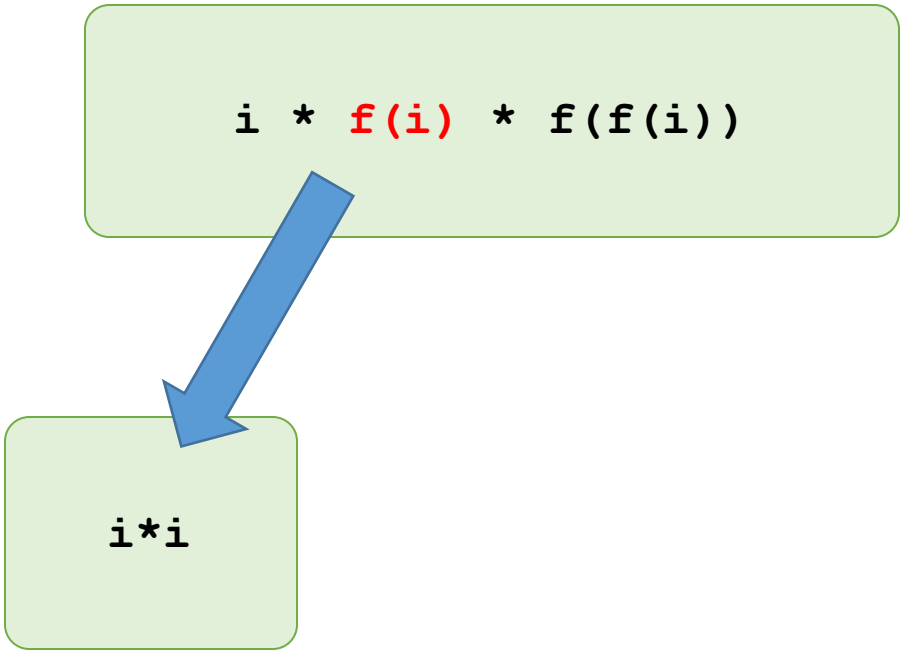
Exercise 3




```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

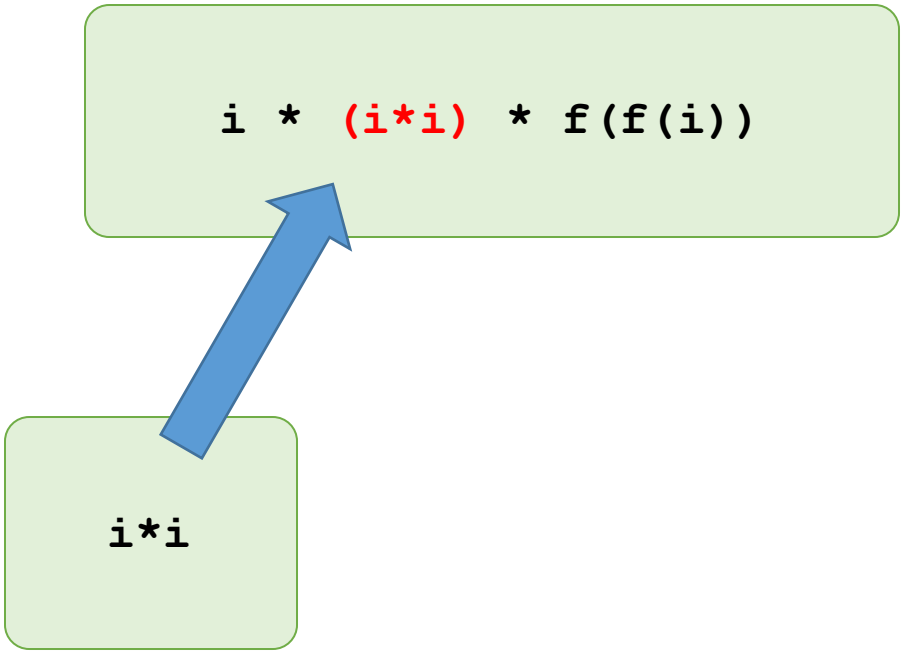
Exercise 3




```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

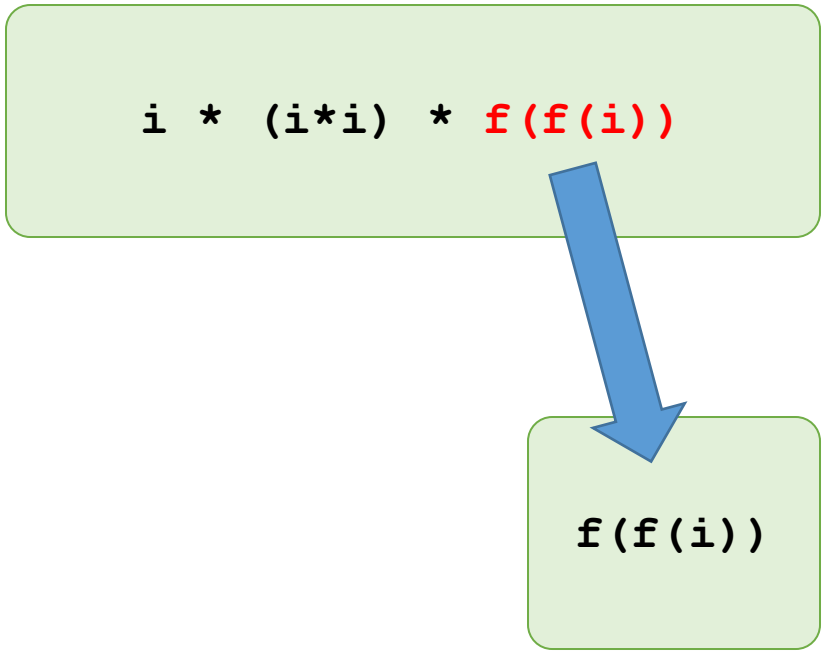
Exercise 3




```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

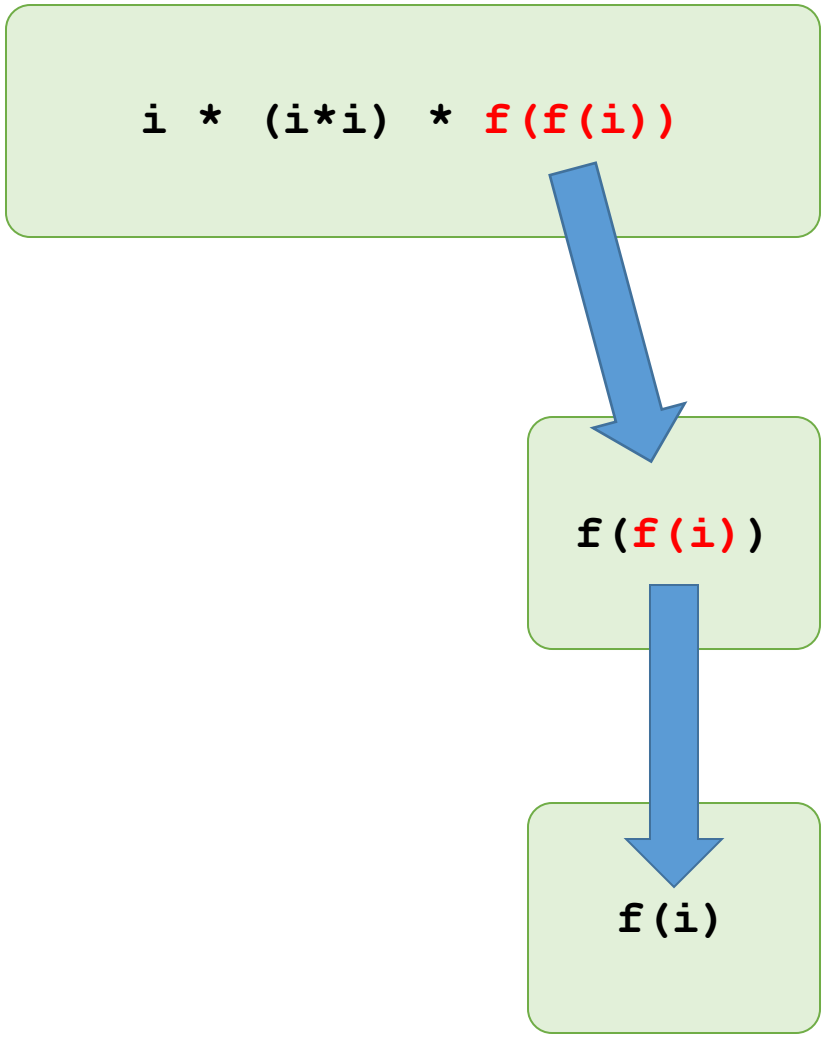
Exercise 3




```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

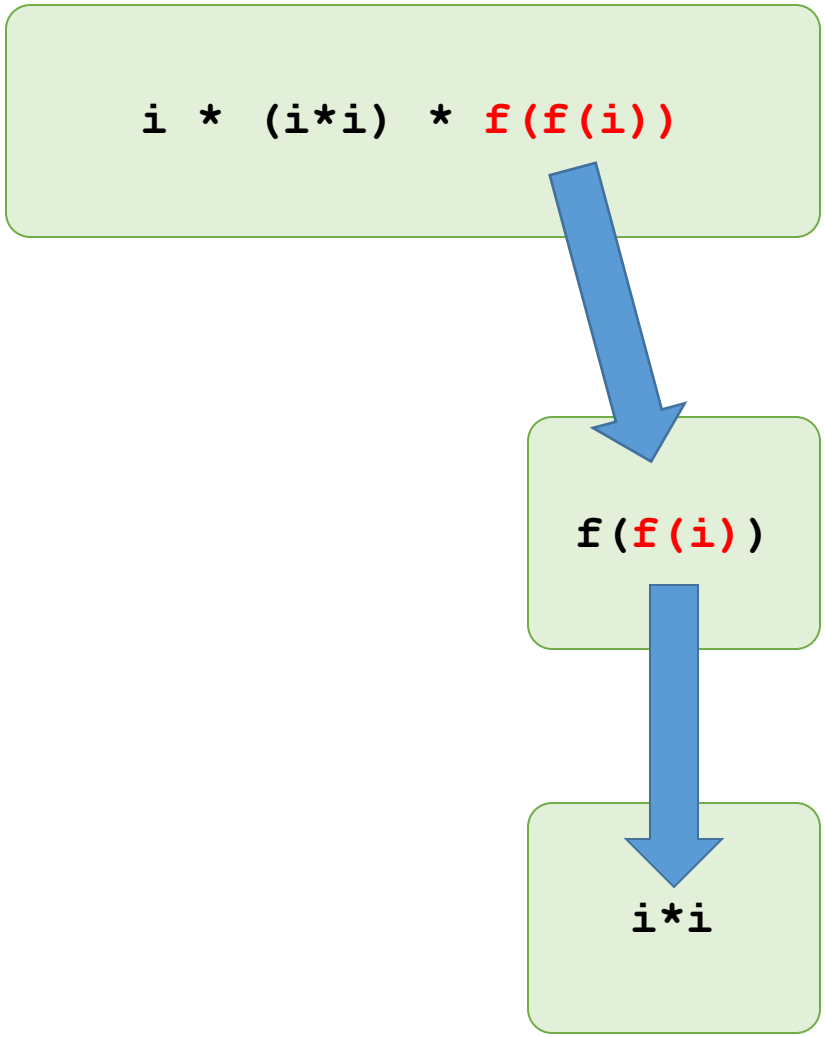
Exercise 3




```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

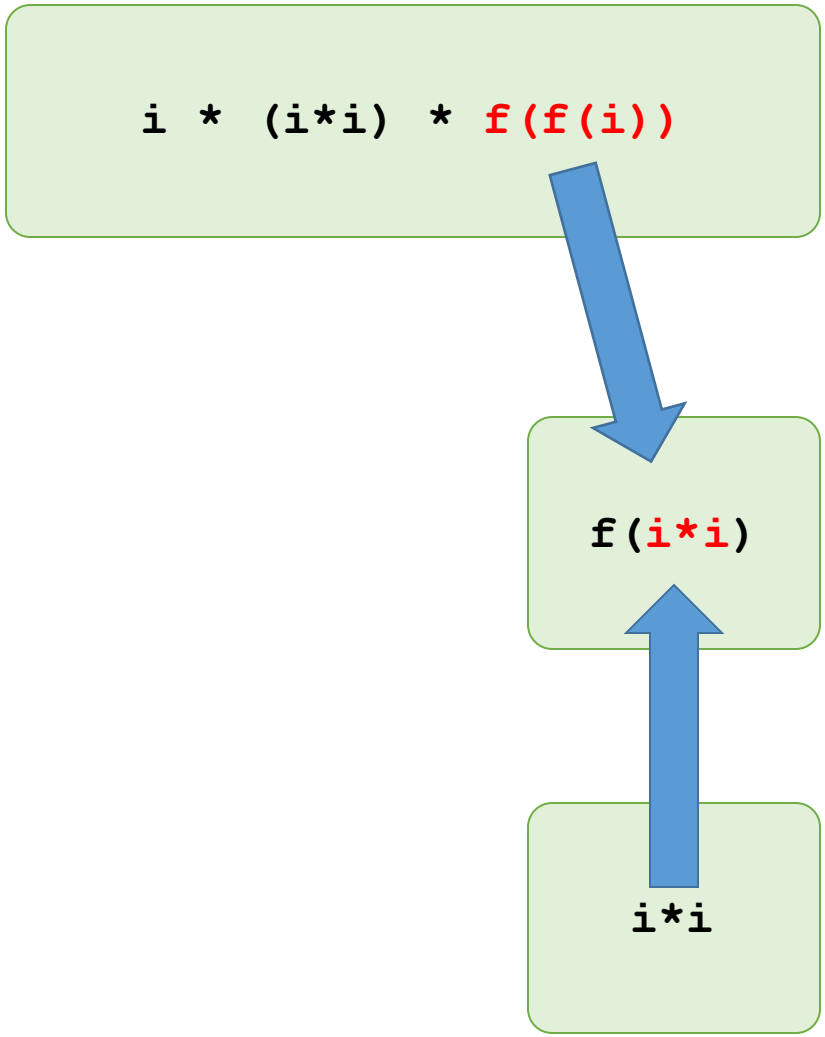
Exercise 3




```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

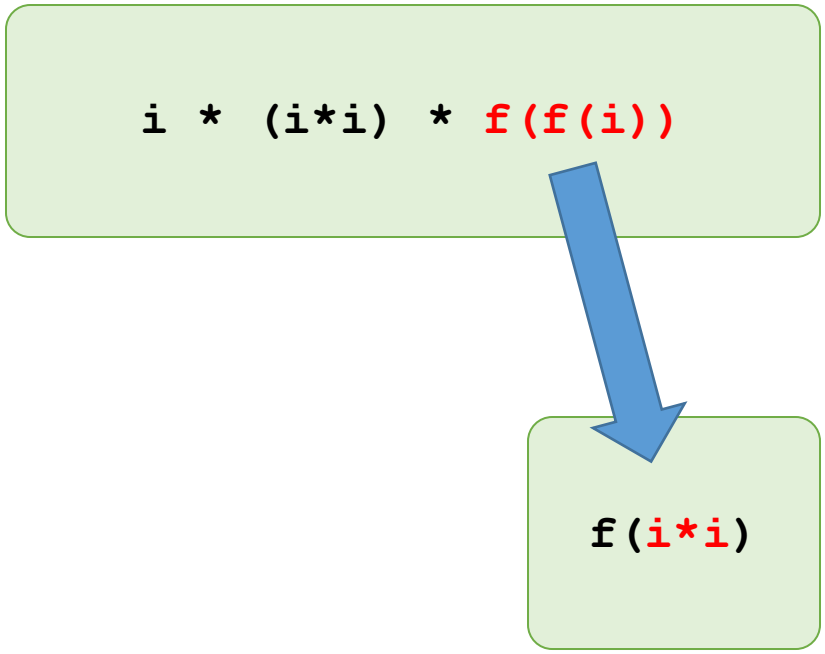
Exercise 3




```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

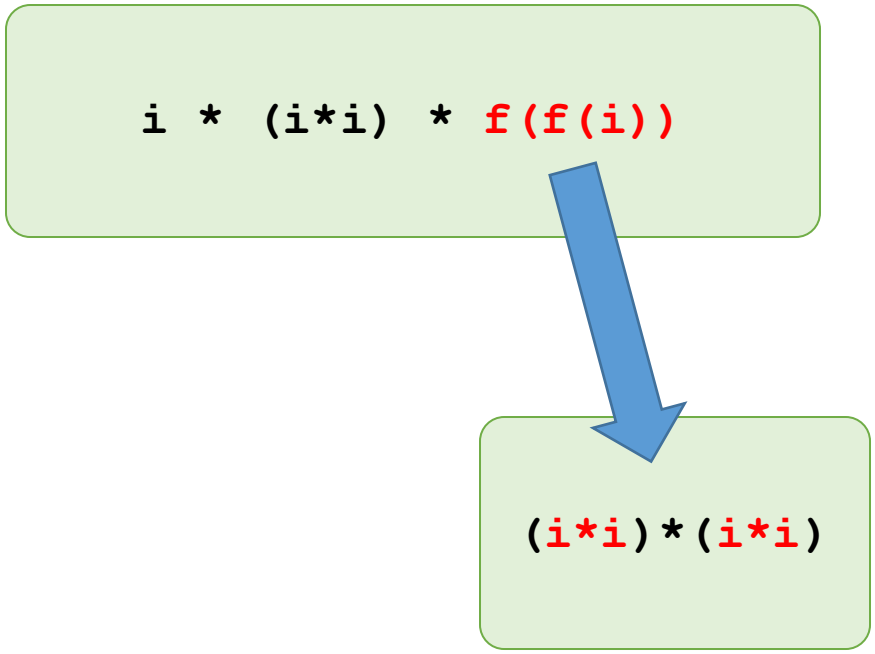
Exercise 3




```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

Exercise 3

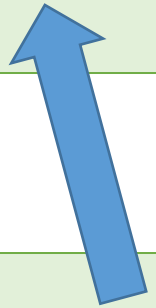



```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

(From: Script Exercise 81)

Exercise 3

```
i * (i*i) * ((i*i)*(i*i))
```



```
(i*i)*(i*i)
```

```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

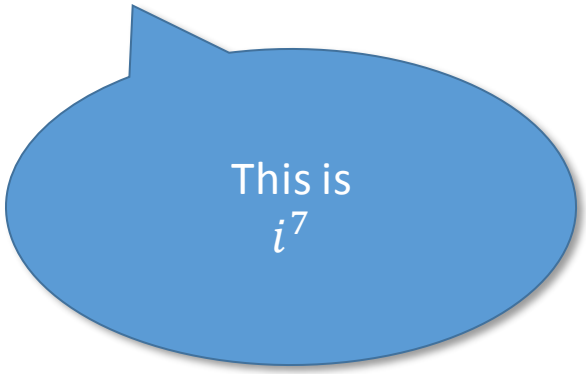
Exercise 3

```
i * (i*i) * ((i*i)*(i*i))
```

```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```

Exercise 3

```
i * (i*i) * ((i*i)*(i*i))
```




```
#include <iostream>

int f (int i) {
     return i * i;
}

int g (int i) {
     return i * f(i) * f(f(i));
}

void h (int i) {
     std::cout << g(i) << "\n";
}

int main () {
     int i;
     std::cin >> i;
     h(i);
     return 0;
}
```