ETH zürich

Informatik

Übungsstunde Woche 6

Pre- and Postconditions

Find **Pre- and Postconditions** 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;
  }
}
```

Pre- and Postconditions

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;
  }
}
```

Pre- and Postconditions

Find **Pre- and Postconditions** 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;
}
```

Pre- and Postconditions

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} and i
<= j
                  // and j < INT_MAX
POST-Condition:         // POST: return value is the sum
                        //          1/i + 1/(i+1) + ... + 1/j
```

Functions

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

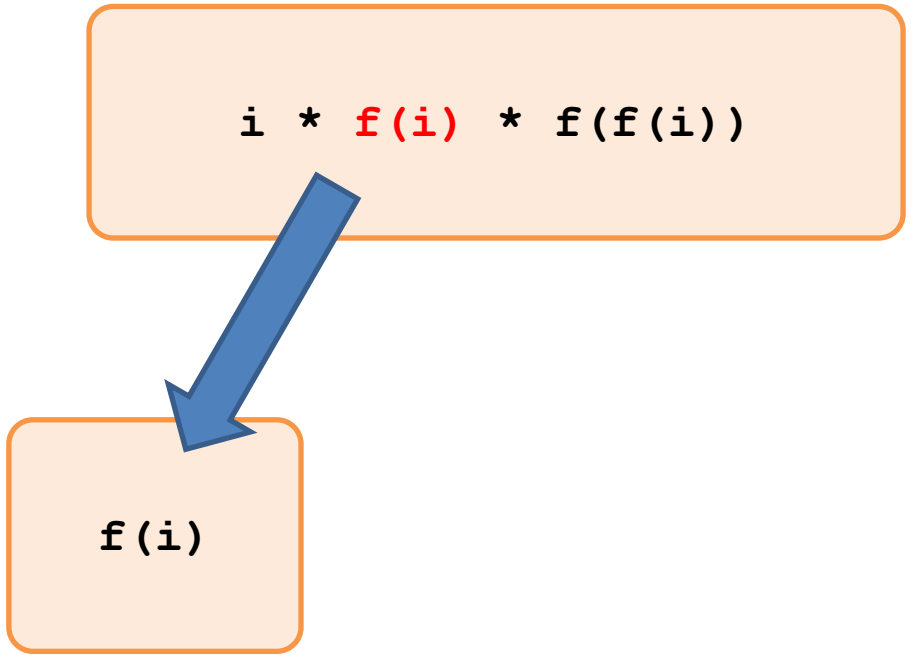
```
#include <iostream>

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;
}
```

Functions

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

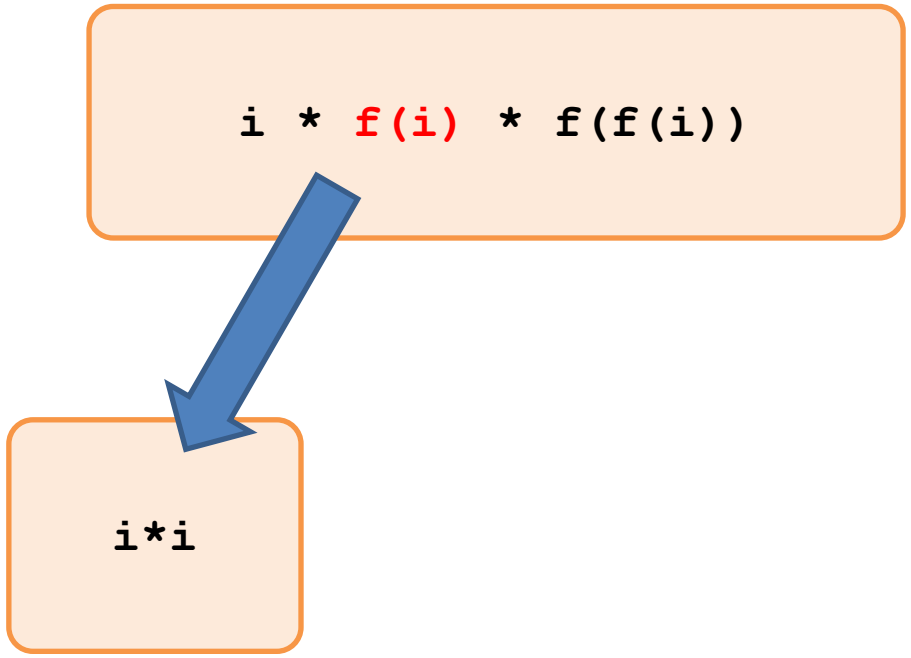
```
#include <iostream>

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;
}
```

Functions



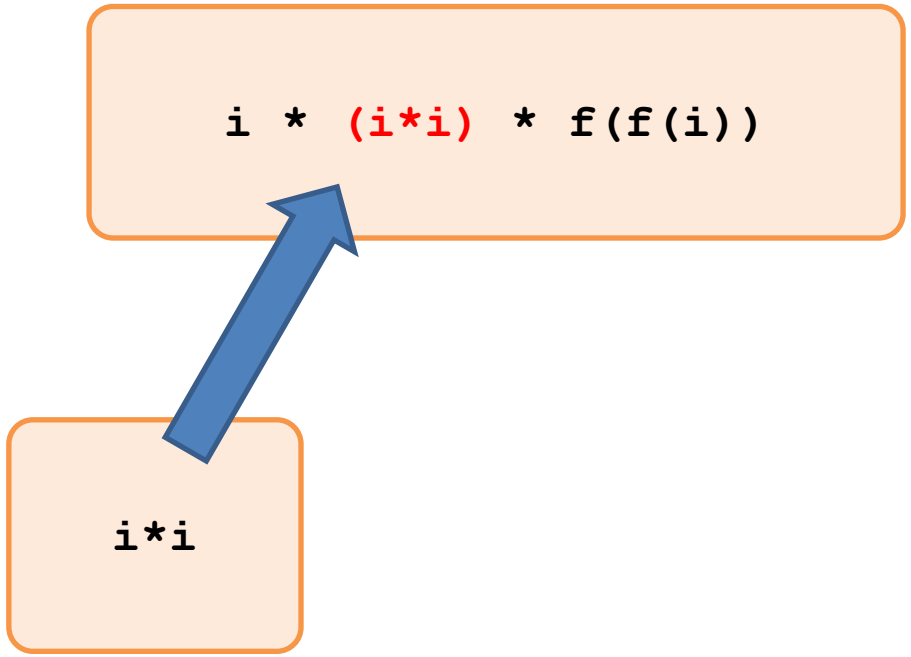

```
#include <iostream>

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;
}
```

Functions



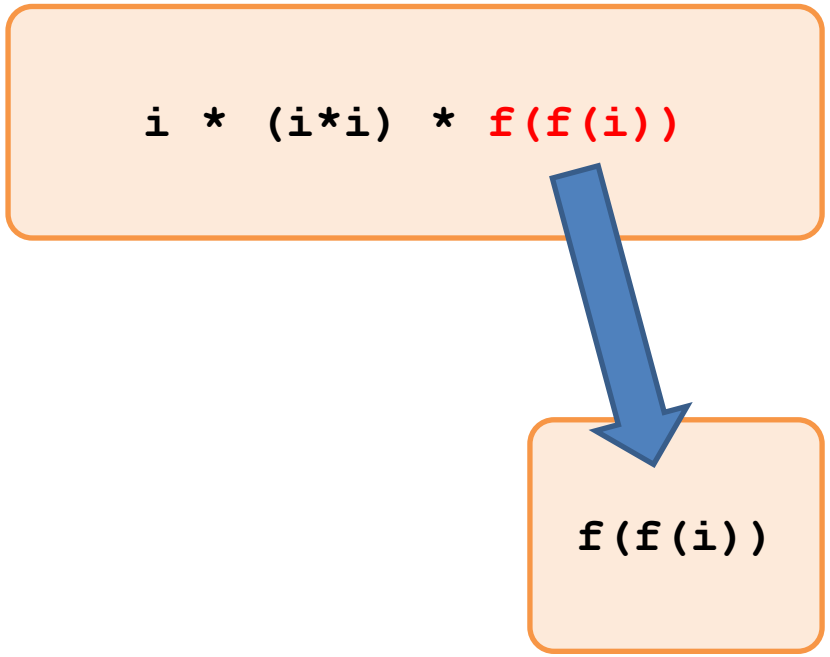

```
#include <iostream>

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;
}
```

Functions



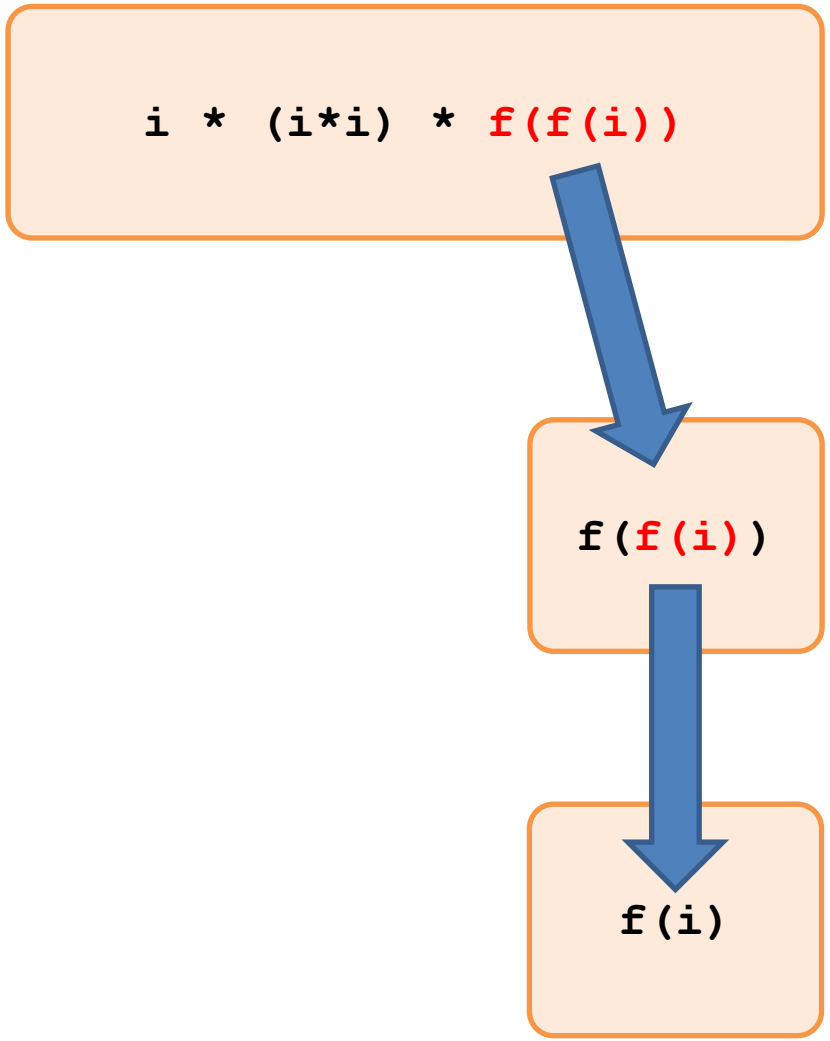

```
#include <iostream>

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;
}
```

Functions



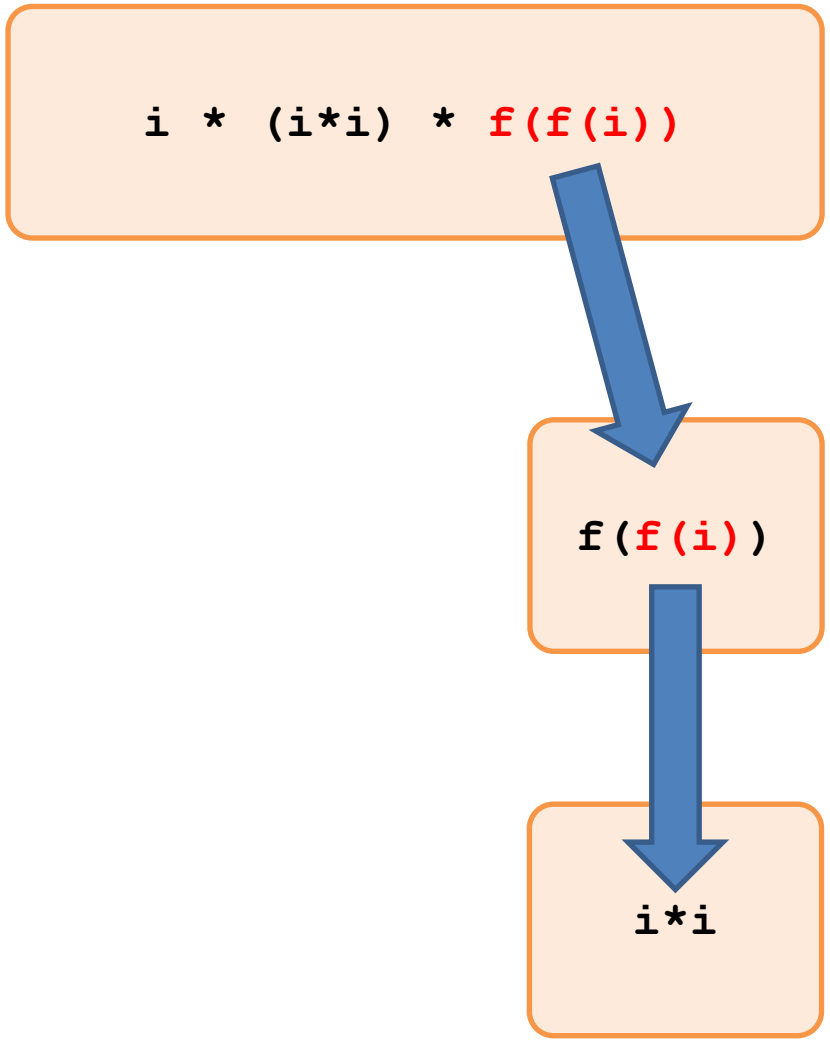

```
#include <iostream>

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;
}
```

Functions



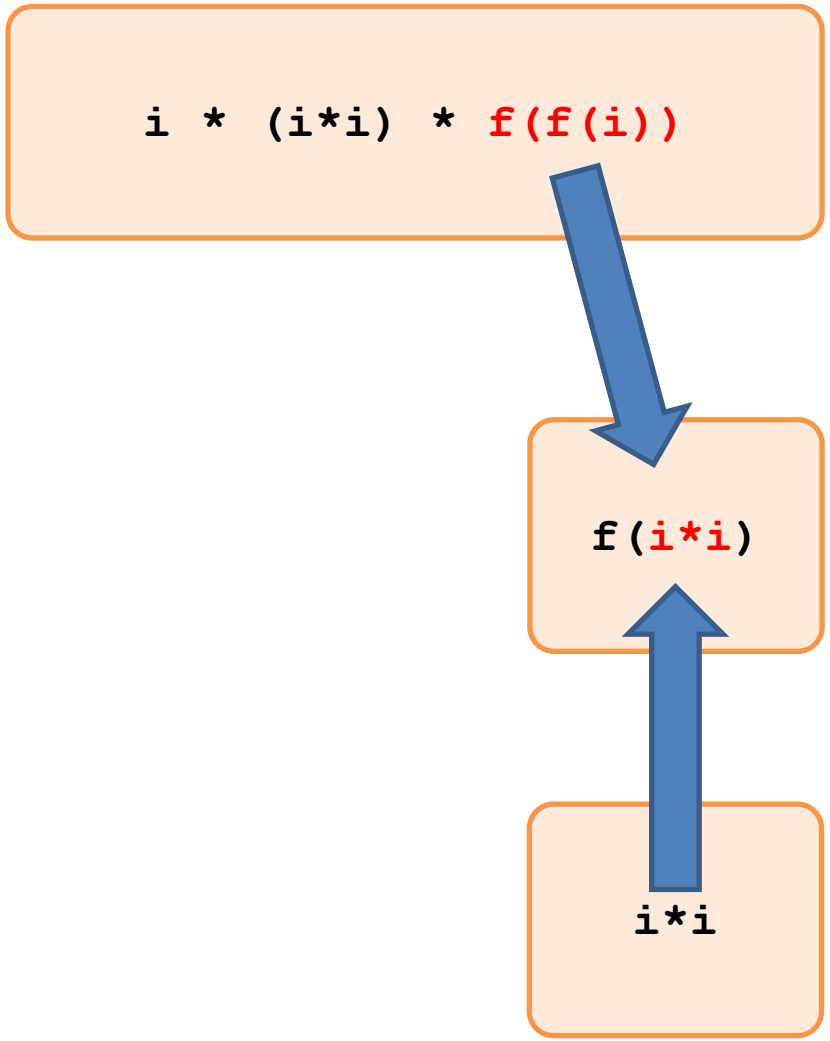

```
#include <iostream>

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;
}
```

Functions



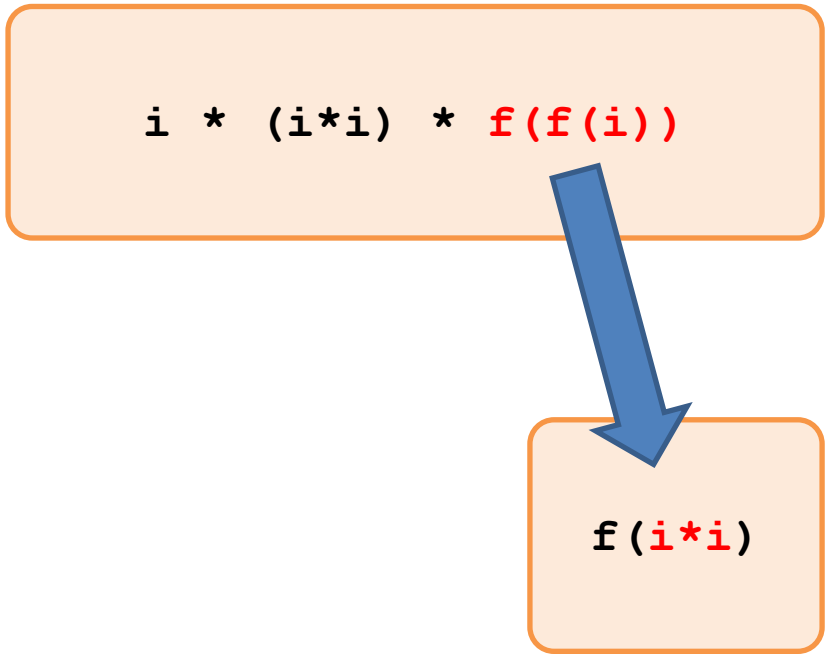

```
#include <iostream>

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;
}
```

Functions




```
#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;
}
```

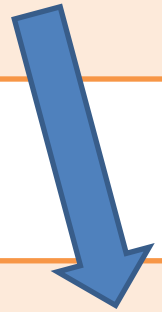
Functions




```
#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;
}
```

Functions

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

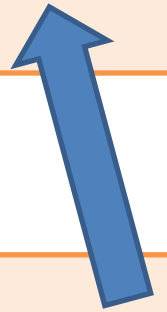


```
(i*i)*(i*i)
```

```
#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;
}
```

Functions

```
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;
}
```

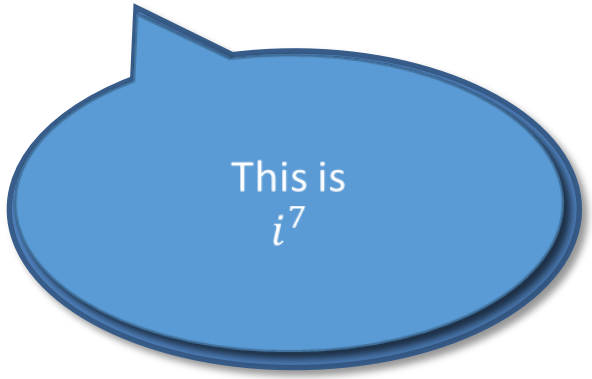
Functions

```
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;
}
```

Functions

```
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;
}
```

Functions

- Find **3 mistakes** in this program.

```
# include <iostream>

double f (double x) {
   return g(2.0 * x);
}

bool g (double x) {
   return x % 2.0 == 0;
}

void h () {
   std::cout << result;
}

int main () {
   double result = f(3.0);
   h();

   return 0;
}
```

Functions

Problem 1: `g()` not yet known

scope of `g` starts later

```
# include <iostream>

double f (double x) {
   return g(2.0 * x);
}

bool g (double x) {
   return x % 2.0 == 0;
}

void h () {
   std::cout << result;
}

int main () {
   double result = f(3.0);
   h();

   return 0;
}
```

Functions

Problem 1: `g()` not yet known

scope of `g` starts later

```
# include <iostream>

double f (double x) {
   return g(2.0 * x);
}

bool g (double x) {
   return x % 2.0 == 0;
}

void h () {
   std::cout << result;
}

int main () {
   double result = f(3.0);
   h();

   return 0;
}
```

Problem 2: Modulo

no modulo for `double`

Functions

Problem 1: `g()` not yet known

scope of `g` starts later

Problem 2: Modulo

no modulo for `double`

Problem 3: `h()` does not «see» `result`

`result` is out-of-scope

```
# include <iostream>

double f (double x) {
   return g(2.0 * x);
}

bool g (double x) {
   return x % 2.0 == 0;
}

void h () {
   std::cout << result;
}

int main () {
   double result = f(3.0);
   h();

   return 0;
}
```

Functions

- Write a function `number_of_divisors` which takes an `int n` as argument and returns the number of divisors of `n` (including `1` and `n`).

```
// PRE: n > 0 and n < MAX_INT
// POST: returns number of divisors of n (incl. 1 and n)
unsigned int number_of_divisors (int n) {
  // your code
}
```

- Example:
  - 6 has 4 divisors, namely 1, 2, 3, 6
    → `std::cout << number_of_divisors(6); // output: 4`

Functions

Solution:

```
// PRE: n > 0 and n < MAX_INT
// POST: returns number of divisors of n (incl. 1 and n)
unsigned int number_of_divisors (int n) {
  assert(n > 0);
  unsigned int counter = 0;
  for (int i = 1; i <= n; ++i)
    if (n % i == 0)
      ++counter;
  return counter;
}
```