Exercise 1
Find **3 mistakes** in this program.

```cpp
#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 1: \( g() \) not yet known

scope of \( g \) starts later
Exercise 1

```cpp
#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 1: `g()` not yet known
Scope of `g` starts later

Problem 2: Modulo
No modulo for `double`
# 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;
}
Exercise 2
Exercise 2

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`).

```cpp
// PRE: n > 0
// 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
/PRE: n > 0
/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;
}