Exercise 1

Exercise 1

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;
}
```

Exercise 1

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;
}
```

Exercise 1

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

Exercise 1

```
# 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

Problem 3: h() does not «see» result

result is out-of-scope

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`).

```
// 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`

Exercise 2

```
// 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;
}
```