Johannes Kepler University Linz
LIT – Secure and Correct Systems Lab
S. Rass · S. Ahmad · M. Schwaighofer ·
K. Feichtinger · D. Zaitsev

PR Systems Programming
WS 2022
Class dates: see KUSSS

Excercise Sheet 2

Warmup (optional): Suppose the following lines are found in the `main()` function. What is wrong here?

a) `printf(enter a number:);`

b) `return "okay";`

c) `void pause() { printf("PAUSE!"); }`

Exercise 2.1. Each of the following programs contains two errors. Find and correct each error.

a)
```
#include<stdio>
int main() {
        // one of Murphy's laws
        print("Whatever can go wrong, will go wrong!\n");
        return 0;
}
```

b)
```
#include<stdio.h>
int main() {
        printf("Whatever can go wrong, will go wrong!"\n)
}
```

c)
```
#include<stdio.h>
int main(int argc, char** argv) {
        / argc counts the command line arguments, and argv contains them /
        printf "we take, but do not use the command line args :-)!\n"
        return 0;
}
```

Exercise 2.2. The standard macro `assert()` checks whether an expression is true (i.e. not equal to 0) or false (i.e. equal to 0).

An example: `assert(a<b);` *`//Program abort if not a < b`*

If the specified expression is false, `assert()` aborts the program and prints an error message containing the expression and the name of the source file with the line number. If the expression is true, the program continues with the next statement.

The `assert()` macro is defined in the `assert.h` header file. It is typically used during the test phase of a program. All assert calls can be deactivated. For this purpose it is sufficient to `#define` the macro `NDEBUG` before `assert.h` is included.

Write a C program that first reads in two floating point numbers from the user. Use the `scanf_s()` function for reading in, if your compiler supports the secure functions, using the `#if ...#else ...#endif` directive. Then divide the first number by the second number and display the result. Make sure with the macro `assert()` that is not divided by zero. Try out the above deactivation mechanism for all assertions.

To write your program, you can use the attached skeleton `assert-skeleton.c`  .

An example output with the assertion failing:

```
Assertion failed!

Program: C:\...\assert-test.exe
File: C:\...\assert-test.c, Line 21

Expression: denominator != 0.0
```