Magdeburg, Nov 24th 2017

Marcus Pinnecke, M.Sc.
Prof. Dr. Gunter Saake

Winter Term 2017/2018

C Language Basics

submission deadline: Dec. 01st 2017 11:59pm

Note: this is a one-week exercise sheet

This is a *per-student exercise* sheet, i.e., you are not allowed to submit a group solution.

You must submit your solution in time via a pull request to the lectures exercise Git repository[1] (cf. sheet No. 1). For this, branch from the official branch `master`, and commit your solutions in a directory `sheet_04/<your last name>`. For your pull request, use the remote branch `submissions` as target for merging.

Prepare to present details of your solution during the tutorium.

Exercise sheet No. 4

After working on this sheet you will have learned the following

✓ The principle of DRY
✓ Recap knowledge on C basics

This sheet consists of the following tasks:

Task 1	**DRY: Don't Repeat Yourself**	2 Point
Task 2	**C Basics: Right or Wrong**	6 Point

Good Luck!

[1] https://github.com/Arcade-Lecture/exercises.git

Task 1 DRY: Don’t Repeat Yourself 2 Point

Don’t Repeat Yourself (DRY) is a principle of software development that tries to reduce the number code clones and, thus, redundancy. DRY states that code cloning for the purpose of enabling slight modifications should be avoided: “every piece of knowledge should occur only once in an entire system”.

(1) Give a code example (favor the C language) that does not follow the principle of DRY!
(2) Give a short description why your code in (1) does not apply the principle of DRY!
(3) Give a DRY code example by modify your code example of (1)!

Provide a textual solution formatted in Markdown in *<repo>*/sheet_04/*<your last name>*/task_1:

Task 2 C Basics: Right or Wrong 6 Point

For each of the following statements, state if the statement is it right or wrong. Feel free to give your statements in this PDF file. Alternatively, create a text file formatted in Markdown containing your solution. Save your solution to *<repo>*/sheet_04/*<your last name>*/task_2.

Id	Statement	Right	Wrong
1	The C language is a high-level programming language that emphasis object-oriented programming.		
2	B.W. Kernighan and D. Ritchie are two of the creators of the C language.		
3	K. Thompson is famous as the inventor of C++.		
4	One of the design goals of C was to create a less complex language in exchange of efficiency in terms of runtime and memory footprint.		
5	Undefined behavior is a behavior where it is not defined how the program behaves (e.g., by illegal memory accesses out of bound, signed integer overflow, null pointer dereference,…)		
6	C is a non-procedural, structured and declarative language with static variable scoping.		
7	In the C language, the concepts of characters, numbers, and addresses are first-class citizens.		
8	The development of UNIX was driven by the need of C in late 1980s.		
9	The **`printf`** function is part of the C core-language.		
10	With the latest changes, a new **`string`** type was added to C (with the specification 2795 as of http://www.faqs.org/rfcs/rfc2795.html).		
11	Object destruction is deterministic in C.		
12	Dynamic memory allocation must be explicitly managed by the programmer.		
13	Dynamic memory deallocation is managed by the C runtime itself.		
14	The preprocessor directive “#force” is part of the C language		
15	The C preprocessor supports file inclusion, macro substitution and conditional complication.		
16	Before C11, every C program based on a single-threaded memory model.		
17	Memory leaks will be no longer a problem once Non-Volatile Random-Access Memory (NVRAM) becomes mainstream.		
18	C11 comes with a rich standard library that a lot of built-in generic (“templated”) data containers (such as linked-lists, arrays, or vectors) and supports multi-threading out-of-the-box.		

19	C’s first implementation traces back before 1989 and was promoted by Brian W. Kernighan und Dennis Ritchie, in their famous book „The C Programming Language”		
20	Bjarne Stroustrup (Creator of C++) is famous for his quote: “C is flexible but neither portable nor often available and often not efficient. This was my main motivation for my new language, the C++ language.”		
21	The following code is a compileable C program in C11 `main() { ; }`		
22	When returning from `main`, the program terminates successfully in case a non-zero value is returned.		
23	Basic input and output (such as file operations) functionality is available due the standard library of C.		
24	A runtime assertion as in `<assert.h>` can be disabled during runtime.		

25	The following code is not a valid C program in C11:
	`#include <stdio.h>`
	`#include <stdbool.h>`
	`#include <iso646.h>`
	`int main(void) { return true or false; }`		

26	The C language nowadays supports atomic data types.		
27	The C programming language is a general-purpose language not only intended for system-level programming but weakly typed (with strong enforcement of weakly types).		
28	Unspecified behavior is behavior where more than one behavior is possible at one instant in time (e.g., order of evaluation). The result of unspecified behavior may differ when repeated.		
29	Implementation-defined behavior is undefined behavior which must be specified and implemented by the programmer.		
30	Single-line comments in C start with #		
31	Single-line comments cannot be used for code documentation.		
32	`/* I believe the compiler ignores all of my comments */`		
33	A type in C is used to interpret a bunch of binary data.		
34	The C type **`byte`** is the smallest addressable value in C.		
35	The word **`bool`** is a keyword in C11, and is used to express a boolean type.		
36	There is support of complex number types in C since C89 from 1989’s.		
37	The types **`short`** and **`short int`** have always the same number of bits in C.		
38	The types **`unsigned int`** is guaranteed to hold have more bits than **`short`**.		
39	The constant `0x23F` is a valid number constant in C.		
40	The constant `0x23L` is a valid number constant in C.		
41	An enumeration tag defines the type for an enumeration in C, e.g., the type of **`enum`** `my_emum` is *my_enum*.		
42	Implicit type conversion take place by the syntax `(type) expression`		
43	Explicit type conversion happens when an expression result differs from the expected type.		
44	When casting integers to floats, there may be the risk of data loss.		
45	When casting characters to integers no data loss can happen.		
46	An object is a piece of memory having a particular value that is the (type-specific) interpretation of that piece of memory.		
47	Padding is alignment of composite types to natural address boundaries to improve memory access performance and must be explicitly turned off.		
48	The string “`2lower`“ is a valid identifier in the C language.		
49	Preprocessor macro names must be always written in capital letters as enumeration constant names must be also written in capital letters.		
50	You are allowed to define the function `void *malloc(size_t, size_t)`.		
51	You are allowed to define the function `const void *return(int)`.		