

Architecting and Engineering Main Memory Database Systems in Modern C

Database and Software Engineering Working Group

Magdeburg, Nov 24th 2017

Winter Term 2017/2018

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

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 repository¹ (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.

This sheet consists of the following tasks:

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

Exercise sheet No. 4

After working on this sheet you will have learned the following

- ✓ The principle of DRY
- ✓ Recap knowledge on C basics

Good Luck!

2 Point

6 Point

¹

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

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.

ld	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.	✓	

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 <stdool.h> #include <stdool.ho< td=""> #inclu</stdool.ho<></stdool.ho<></stdool.ho<></stdool.ho<></stdool.ho<></stdool.ho<></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></assert.h>	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	✓	
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 wain() { ; } When returning from main, the program terminates successfully in case a non-zero value is returned. Basic input and output (such as file operations) functionality is available due the standard library of C. 44 A runtime assertion as in -assert.h- can be disabled during runtime. The following code is not a valid C program in C11: #include <stdio.h-> #inclu</stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h-></stdio.h->				
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.he can be disabled during runtime. 25 The following code is not a valid C program in C11: #include <stdbool.h> #include <stdbool.h> #include <stdbool.h> #include <stdool.h> #include <stdool.hol #="" (e.g.,="" (with="" *="" 1989's.="" 26="" 27="" 28="" 29="" 30="" 31="" 32="" 33="" 34="" 35="" 36="" 37="" 38="" a="" addressable="" all="" always="" and="" at="" atomic="" be="" behavior="" believe="" binary="" bits="" bool.="" boolean="" bunch="" but="" by="" byte="" c="" c.="" c11,="" c89="" cannot="" code="" comments="" compiler="" complex="" data="" data.="" differ="" documentation.="" enforcement="" evaluation).="" express="" for="" from="" general-purpose="" i="" ignores="" implementation-defined="" implemented="" in="" instant="" intended="" interpret="" is="" keyword="" language="" may="" more="" must="" my="" not="" nowadays="" number="" of="" of<="" one="" only="" order="" possible="" programmer.="" programming="" repeated.="" result="" same="" short="" since="" single-line="" smallest="" specified="" start="" strong="" support="" supports="" system-level="" td="" than="" thave="" the="" there="" time="" to="" type="" type.="" typed="" types="" types).="" types.="" undefined="" unspecified="" used="" value="" weakly="" when="" where="" which="" with="" word=""><td>20</td><td>neither portable nor often available and often not efficient. This was my</td><td></td><td>✓</td></stdool.hol></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdbool.h></stdbool.h></stdbool.h>	20	neither portable nor often available and often not efficient. This was my		✓
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 <stdio.h> #include <stdio.h> #include <stdio.h> #include <stode.h #="" (e.g.,="" (with="" 0x:231="" 1989's.="" 26="" 27="" 28="" 29="" 30="" 31="" 33="" 34="" 35="" 36="" 37="" 38="" 39="" 40="" 41="" 42="" 44="" 45="" 46="" a="" addressable="" always="" an="" and="" at="" atomic="" be="" behavior="" binary="" bits="" bool="" boolean="" bunch="" but="" by="" byte="" c="" c,="" c.="" c11,="" c89="" can="" casting="" characters="" comments="" complex="" constant="" conversion="" data="" data.="" defines="" differ="" differs="" e.g.,="" enforcement="" enum="" enumeration="" evaluation).="" expected="" express="" expression="" false;="" floats,="" for="" from="" general-purpose="" happen.="" happens="" have="" havi<="" implementation-defined="" implemented="" implicit="" in="" instant="" int="" integers="" intended="" interpret="" is="" keyword="" language="" loss="" loss.="" main(void)="" may="" memory="" more="" must="" my_enum="" my_enum.="" no="" not="" nowadays="" number="" object="" of="" one="" only="" or="" order="" piece="" possible="" programmer.="" programming="" repeated.="" result="" return="" risk="" same="" short="" since="" single-line="" smallest="" specified="" start="" strong="" support="" supports="" system-level="" tag="" td="" than="" the="" there="" time="" to="" true="" type="" type.="" typed="" types="" types).="" types.="" undefined="" unspecified="" used="" valid="" value="" weakly="" when="" where="" which="" with="" word="" {="" }=""><td>21</td><td></td><td></td><td>. /</td></stode.h></stdio.h></stdio.h></stdio.h></stdio.h>	21			. /
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 <stdoio.h> #include <stdoio.< td=""><td></td><td></td><td></td><td> </td></stdoio.<></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></stdoio.h></assert.h>				
the standard library of C. 4 A runtime assertion as in <assert.h> can be disabled during runtime. 5 The following code is not a valid C program in C11: #include <stdio.h> #include <stdio.h> #include <stdool.h> #include <stdool.h #include="" <st<="" <stdool.h="" td=""><td>22</td><td></td><td></td><td>✓</td></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdool.h></stdio.h></stdio.h></assert.h>	22			✓
The following code is not a valid C program in C11: #include <stdio.h> #include <stdio.h> #include <stdioo.h> #include <stdioo.h> #include <stdiool.h> #include <stdiool.h #="" (e.g.,="" (type)="" (with="" *="" 0x23f="" 1989's.="" 27="" 28="" 29="" 30="" 31="" 32="" 33="" 34="" 35="" 36="" 37="" 38="" 39="" 40="" 41="" 42="" 43="" 44="" 45="" a="" addressable="" all="" allways="" an="" and="" at="" atomic="" be="" behavior="" believe="" binary="" bits="" bool="" boolean="" bunch="" but="" by="" byte="" c="" c,="" c.="" c11,="" c89="" can="" cannot="" casting="" characters="" code="" comments="" compiler="" complex="" constant="" conversion="" data="" data.="" defines="" differ="" documentation.="" e.g.,="" enforcement="" enum="" enum.="" enumeration="" evaluation).="" explicit="" express="" expression="" for="" from="" general-purpose="" guaranteed="" happen.="" have="" hold="" i="" ignores="" implementation-defined="" implemented="" implicit="" in="" instant="" int="" integers="" intended="" interpret="" is="" keyword="" language="" los<="" loss="" may="" more="" must="" my="" no="" not="" nowadays="" number="" of="" one="" only="" order="" place="" possible="" programmer.="" programming="" repeated.="" result="" same="" short="" short.="" since="" single-line="" smallest="" specified="" start="" strong="" support="" supports="" syntax="" system-level="" tag="" take="" td="" than="" the="" there="" time="" to="" type="" type.="" typed="" types="" types).="" types.="" undefined="" unsigned="" unspecified="" used="" valid="" value="" weakly="" when="" where="" which="" with="" word=""><td>23</td><td>, , , , , , , , , , , , , , , , , , , ,</td><td>✓</td><td></td></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdiool.h></stdioo.h></stdioo.h></stdio.h></stdio.h>	23	, , , , , , , , , , , , , , , , , , , ,	✓	
#include <stdbool.h> #include <sto646.h> int main(void) { return true or false; } 27 The C language nowadays supports atomic data 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 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 0x23F 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 enum is my enum. 42 Implicit type conversion happens when an expression result differs from the expected type. 43 Explicit type conversion happens when an expression result differs from the expected type. 44 When casting characters to integers no data loss can happen. 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 Preprocessor macro names must be always written in capital letters a enumeration constant names mu</sto646.h></stdbool.h></stdbool.h></stdbool.h></stdbool.h></stdbool.h></stdbool.h></stdbool.h></stdbool.h></stdbool.h>	24	A runtime assertion as in <assert.h> can be disabled during runtime.</assert.h>		✓
#include <iscdbool.h> #include <iscdbool.h> #include <iscdbool.h> #include <iscdbool.h> #include <iscdbool.h> #include <iscdbool.h> #include <iscdbool.ho #incl<="" #include="" <iscdbool.ho ="" td=""><td>25</td><td>The following code is not a valid C program in C11:</td><td></td><td></td></iscdbool.ho ></iscdbool.h></iscdbool.h></iscdbool.h></iscdbool.h></iscdbool.h></iscdbool.h>	25	The following code is not a valid C program in C11:		
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 betieve the compiter 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 enum 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.		<pre>#include <stdbool.h> #include <iso646.h></iso646.h></stdbool.h></pre>	✓	
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 in C start with # 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_enum is my_enum. 42 Implicit 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 integers to floats, there may be the risk of data loss. 46 An object is a piece of memory having a particular value that is the (typespecific) 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.	26	The C language nowadays supports atomic data types.	✓	
at one instant in time (e.g., order of evaluation). The result of unspecified behavior may differ when repeated. Implementation-defined behavior is undefined behavior which must be specified and implemented by the programmer. Single-line comments in C start with # Single-line comments cannot be used for code documentation. 2 /* I betieve the compiler ignores all of my comments */ A type in C is used to interpret a bunch of binary data. The C type byte is the smallest addressable value in C. The word bool is a keyword in C11, and is used to express a boolean type. The types short and short int have always the same number of bits in C. The types unsigned int is guaranteed to hold have more bits than short. The constant 0x23F is a valid number constant in C. The constant 0x23L is a valid number constant in C. An enumeration tag defines the type for an enumeration in C, e.g., the type of enum my_enum is my_enum. Implicit type conversion happens when an expression result differs from the expected type. When casting integers to floats, there may be the risk of data loss. When casting characters to integers no data loss can happen. An object is a piece of memory having a particular value that is the (type-specific) interpretation of that piece of memory. Padding is alignment of composite types to natural address boundaries to improve memory access performance and must be explicitly turned off. When casting "21ower" is a valid identifier in the C language. Preprocessor macro names must be always written in capital letters as enumeration constant names must be always written in capital letters.	27	intended for system-level programming but weakly typed (with strong enforcement of weakly types).	✓	
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 0x23F 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.	28	at one instant in time (e.g., order of evaluation). The result of unspecified	✓	
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_enum 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 "2\tower" 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.		specified and implemented by the programmer.	✓	
32				
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_enum 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 integers 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 "21ower" 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.			,	
The C type byte is the smallest addressable value in C. The word bool is a keyword in C11, and is used to express a boolean type. The types short and short int have always the same number of bits in C. The types unsigned int is guaranteed to hold have more bits than short. The constant 0x23F is a valid number constant in C. The constant 0x23L is a valid number constant in C. The constant 0x23L is a valid number constant in C. The constant 0x23L is a valid number constant in C. Inplicit type conversion take place by the syntax (type) expression Explicit type conversion happens when an expression result differs from the expected type. When casting integers to floats, there may be the risk of data loss. When casting characters to integers no data loss can happen. An object is a piece of memory having a particular value that is the (type-specific) interpretation of that piece of memory. Padding is alignment of composite types to natural address boundaries to improve memory access performance and must be explicitly turned off. The string "21ower" is a valid identifier in the C language. Preprocessor macro names must be also written in capital letters as enumeration constant names must be also written in capital letters.	\vdash			
The word bool is a keyword in C11, and is used to express a boolean type. The is support of complex number types in C since C89 from 1989's. The types short and short int have always the same number of bits in C. The types unsigned int is guaranteed to hold have more bits than short. The constant 0x23F is a valid number constant in C. The constant 0x23L is a valid number constant in C. The constant 0x23L is a valid number constant in C. In an enumeration tag defines the type for an enumeration in C, e.g., the type of enum my_enum is my_enum. Implicit type conversion take place by the syntax (type) expression Implicit type conversion happens when an expression result differs from the expected type. When casting integers to floats, there may be the risk of data loss. When casting characters to integers no data loss can happen. An object is a piece of memory having a particular value that is the (type-specific) interpretation of that piece of memory. Padding is alignment of composite types to natural address boundaries to improve memory access performance and must be explicitly turned off. The string "2lower" is a valid identifier in the C language. Preprocessor macro names must be always written in capital letters as enumeration constant names must be also written in capital letters.				. /
There is support of complex number types in C since C89 from 1989's. The types short and short int have always the same number of bits in C. The types unsigned int is guaranteed to hold have more bits than short. The constant 0x23F is a valid number constant in C. The constant 0x23L is a valid number constant in C. The constant 0x23L is a valid number constant in C. In an enumeration tag defines the type for an enumeration in C, e.g., the type of enum my_emum is my_enum. Implicit type conversion take place by the syntax (type) expression Explicit type conversion happens when an expression result differs from the expected type. When casting integers to floats, there may be the risk of data loss. When casting characters to integers no data loss can happen. An object is a piece of memory having a particular value that is the (type-specific) interpretation of that piece of memory. Padding is alignment of composite types to natural address boundaries to improve memory access performance and must be explicitly turned off. The string "2lower" is a valid identifier in the C language. Preprocessor macro names must be always written in capital letters as enumeration constant names must be also written in capital letters.			1/	_ <u> </u>
The types short and short int have always the same number of bits in C. The types unsigned int is guaranteed to hold have more bits than short. The constant 0x23F is a valid number constant in C. The constant 0x23L is a valid number constant in C. An enumeration tag defines the type for an enumeration in C, e.g., the type of enum my_emum is my_enum. Implicit type conversion take place by the syntax (type) expression Explicit type conversion happens when an expression result differs from the expected type. When casting integers to floats, there may be the risk of data loss. When casting characters to integers no data loss can happen. An object is a piece of memory having a particular value that is the (type-specific) interpretation of that piece of memory. Padding is alignment of composite types to natural address boundaries to improve memory access performance and must be explicitly turned off. The string "2lower" is a valid identifier in the C language. Preprocessor macro names must be always written in capital letters as enumeration constant names must be also written in capital letters.			· ·	1
The types unsigned int is guaranteed to hold have more bits than short. The constant 0x23F is a valid number constant in C. The constant 0x23L is a valid number constant in C. An enumeration tag defines the type for an enumeration in C, e.g., the type of enum my_emum is my_enum. Implicit type conversion take place by the syntax (type) expression Explicit type conversion happens when an expression result differs from the expected type. When casting integers to floats, there may be the risk of data loss. When casting characters to integers no data loss can happen. An object is a piece of memory having a particular value that is the (type-specific) interpretation of that piece of memory. Padding is alignment of composite types to natural address boundaries to improve memory access performance and must be explicitly turned off. The string "21ower" is a valid identifier in the C language. Preprocessor macro names must be always written in capital letters as enumeration constant names must be also written in capital letters.		7, 7,		-
The constant 0x23F is a valid number constant in C. The constant 0x23L is a valid number constant in C. An enumeration tag defines the type for an enumeration in C, e.g., the type of enum my_emum is my_enum. Implicit type conversion take place by the syntax (type) expression Explicit type conversion happens when an expression result differs from the expected type. When casting integers to floats, there may be the risk of data loss. When casting characters to integers no data loss can happen. An object is a piece of memory having a particular value that is the (type-specific) interpretation of that piece of memory. Padding is alignment of composite types to natural address boundaries to improve memory access performance and must be explicitly turned off. The string "21ower" is a valid identifier in the C language. Preprocessor macro names must be always written in capital letters as enumeration constant names must be also written in capital letters.				. /
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 "21ower" 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.			✓	
of enum my_enum 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.	40	The constant 0x23L is a valid number constant in C.		
Implicit type conversion take place by the syntax (type) expression V	41	• • • • • • • • • • • • • • • • • • • •		
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.			'	
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.	-			V
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 (typespecific) 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.		expected type.		
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.	-		,	_
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.	-		✓	
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.		specific) interpretation of that piece of memory.	✓	
48 The string "21ower" 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.	47		✓	
Preprocessor macro names must be always written in capital letters as enumeration constant names must be also written in capital letters.	10			
enumeration constant names must be also written in capital letters.				
	49	·		
TOO TION ALO MILOWOU TO MOITHO THE INFINITION FULL AMOUNT AND THE CONTRACT OF	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).			,	/