MISRA-C:2012(Ed 3 Rev 1) Standards Model Summary for C / C++

The LDRA tool suite® is developed and certified to BS EN ISO 9001:2015, TÜV SÜD and SGS-TÜV Saar.

This information is applicable to version 9.8.2 of the LDRA tool suite®. It is correct as of 16th April 2019.

© Copyright 2019 LDRA Ltd. All rights reserved.

Compliance is measured against

"MISRA C:2012 Guidelines for the use of the C language in critical systems (Third Edition, first revision)" 2019

Copyright © MISRA

Further information is available at http://www.misra.org.uk

Classification	Enhanced Enforcement	Fully Implemented	Partially Implemen ted	Imniamantad	Not statically Checkable	Total
Mandatory	0	12	4	0	0	16
Required	10	95	11	0	2	118
Advisory	7	26	6	0	0	39
Total	17	133	21	0	2	173

	MISRA-C:2012(Ed 3 Rev 1) Standards Model Compliance for C / C++				
Rule	Classification	Rule Description	LDRA Standard	LDRA Standard Description	
D.1.1	Required	Any implementation-defined behaviour on which	69 S	#pragma used.	
D.1.1	Required	the output of the program depends shall be	584 S	Remainder of % op could be negative.	
D.2.1	Required	All source files shall compile without any compilation errors			
D.3.1	Required	All code shall be traceable to documented requirements			
			43 D	Divide by zero found.	
				Pointer not checked for null before use.	
			115 D	Copy length parameter not checked before use.	
			123 D	File pointer not checked for null before use.	
			127 D	Local or member denominator not checked before use.	
	Required	Run-time failures shall be minimised	128 D	Global pointer not checked within this procedure.	
			129 D	Global file pointer not checked within this procedure.	
D.4.1			131 D	Global denominator not checked within this procedure.	
			135 D	Pointer assigned to NULL may be dereferenced.	
			136 D	Global pointer assigned to NULL may be dereferenced.	
			137 D	Parameter used as denominator not checked before use.	
				Divide by zero in preprocessor directive.	
				Numeric overflow.	
				Numeric underflow.	
				Divide by zero found.	
			80 X	Divide by zero found.	
D.4.2	Advisory	All usage of assembly language should be documented	17 S	Code insert found.	
D.4.3	Required	Assembly language shall be encapsulated and isolated	88 S	Procedure is not pure assembler.	
D.4.4	Advisory	Sections of code should not be 'commented out'	302 S	Comment possibly contains code.	

		Identifiers in the same namespace with	247.0	Names only differ by case.
D.4.5	Advisory	overlapping visibility should be typographically	217 S 67 X	
		typedefs that indicate size and signedness should	90 S	Identifier is typographically ambiguous.
D.4.6	Advisory	171		Basic type declaration used.
		be used in place of the basic numerical types	495 S	Typedef name has no size indication.
			91 D	Function return value potentially unused. Var set by std lib func return not checked
D.4.7	Required	If a function returns error information, then that	124 D	before use.
D. 1	rtoquirou	error information shall be tested		Global set by std lib func return not checked
			130 D	before use.
		If a pointer to a structure or union is never		
D.4.8	Advisory	dereferenced within a translation unit, then the	104 D	Structure implementation not hidden.
		implementation of the object should be hidden		
		A function should be used in preference to a		
D.4.9	Advisory	function-like macro where they are	340 S	Use of function like macro.
		interchangeable		
5		Precautions shall be taken in order to prevent the	242.0	
D.4.10	Required	contents of a header file being included more than	243 S	Included file not protected with #define.
		Once		
D.4.11	Required	The validity of values passed to library functions shall be checked		
D.4.12	Required	Dynamic memory allocation shall not be used	44 S	Use of banned function, type or variable.
D.4.12	Required	Functions which are designed to provide	44 3	Ose of barriled function, type of variable.
D.4.13	Advisory	operations on a resource should be called in an		
D.4.10	Advisory	appropriate sequence		
		appropriate sequence	43 D	Divide by zero found.
			45 D	Pointer not checked for null before use.
			85 D	Filename not verified before fopen.
			86 D	User input not checked before use.
			123 D	File pointer not checked for null before use.
			127 D	Local or member denominator not checked
			127 D	before use.
			128 D	Global pointer not checked within this
D.4.14	Required	The validity of values received from external	120 D	procedure.
D.4.14	Required	sources shall be checked	129 D	Global file pointer not checked within this
			120 2	procedure.
			131 D	Global denominator not checked within this
				procedure.
			248 S	Divide by zero in preprocessor directive.
			493 S	Numeric overflow.
			494 S	Numeric underflow.
			629 S	Divide by zero found.
			80 X	Divide by zero found.

			21 S	
			145 S	#if has invalid expression.
			323 S	Switch has more than one default case.
			345 S	Bit operator with floating point operand.
		The program shall contain no violations of the	387 S	Enum init not integer-constant-expression.
R.1.1	Required	standard C syntax and constraints, and shall not	404 S	Array initialisation has too many items.
		exceed the implementation's translation limits	481 S	Array with no bounds in struct.
			580 S	Macro redefinition without using #undef.
			615 S	Conditional operator has incompatible types.
			646 S	Struct initialisation has too many items.
			110 S	Use of single line comment //.
			143 S	Curly brackets used in expression.
R.1.2	Advisory	Language extensions should not be used	293 S	Non ANSI/ISO construct used.
			632 S	Use of // comment in pre-processor directive
			032 3	or macro defn.
R.1.3	Required	There shall be no occurrence of undefined or critical unspecified behaviour	82 D	fsetpos values not generated by fgetpos.
			83 D	<i>y</i> 1
				No fseek or flush before I/O.
				Illegal shared object in signal handler.
			89 D	9
				File does not end with new line.
				Number of parameters does not match.
				Use of banned function, type or variable.
				Void procedure used in expression.
				Void variable passed as parameter.
				Non standard character in source.
			118 S	main must be int (void) or int (int,char*[]).
				Non standard escape sequence in source.
			296 S	Function declared at block scope.
			324 S	
			335 S	Operator defined contains illegal items.
			336 S	
				Undefined behaviour, \ before E-O-F.
				Wide string and string concatenated.
			465 S	Struct/union not completely specified.
			482 S	
			486 S	Incorrect number of formats in output function.

D 4 2 (acations all			407.0	Inquifficient energy allocated
R.1.3 (continued)				Insufficient space allocated.
			489 S	Insufficient space for operation.
				Type is incomplete in translation unit.
			573 S	Macro concatenation of uni char names.
			576 S	Function pointer is of wrong type.
			582 S	const object reassigned.
			587 S	Const local variable not immediately initialised.
			589 S	Format is not appropriate type.
			590 S	Mode fault in fopen.
			608 S	Use of explicitly undefined language feature.
			642 S	Function return type with array field.
			645 S	realloc ptr type does not match target type.
			649 S	Use of unallocated flexible array.
				Flexible array copy ignores last member.
			66 X	Insufficient array space at call.
			70 X	Array has insufficient space.
				Insufficient space for copy.
			79 X	Size mismatch in memcpy/memset.
			28 D	Potentially infinite loop found.
			76 D	Procedure is not called or referenced in code analysed.
R.2.1	Required	A project shall not contain unreachable code	1 J	Unreachable Code found.
	•		3 J	All internal linkage calls unreachable.
			35 S	Static procedure is not explicitly called in code analysed.
			8 D	DD data flow anomalies found.
			65 D	Void function has no side effects.
R.2.2	Required	There shall be no dead code	105 D	DU anomaly dead code, var value is unused on all paths.
			57 S	Statement with no side effect.
R.2.3	Advisory	A project should not contain unused type declarations	413 S	User type declared but not used in code analysed.
R.2.4	Advisory	A project should not contain unused tag declarations	413 S	User type declared but not used in code analysed.
R.2.5	Advisory	A project should not contain unused macro declarations	628 S	Macro not used in translation unit.

R.2.6 Advisory declarations R.2.7 Advisory declarations R.2.7 Advisory declarations There should be no unused parameters in functions The character sequences /* and // shall not be used procedure parameter. The character sequences /* and // shall not be used procedure parameter. The character sequences /* and // shall not be used procedure parameter. The character sequences /* and // shall not be used in // comment found. R.3.2 Required Line-splicing shall not be used in // comments Octal and hexadecimal escape sequences shall be trained to the state of the same sequences shall be trained to the same sequences shall be trained to the same sequences shall be trained to the same sequence in the					1
R.3.1 Required The character sequences /* and // shall not be used within a comment used within a comment to used within a comment to Catal and hexadecimal escape sequences shall be terminated to Catal and hexadecimal escape sequences shall be terminated to Catal and hexadecimal escape sequences shall be terminated to Catal and hexadecimal escape sequences shall be terminated to Catal and hexadecimal escape sequences shall be terminated to Catal and hexadecimal escape sequences shall be terminated to Catal and hexadecimal escape sequences shall be terminated to Catal and hexadecimal escape sequences shall be terminated to Catal and hexadecimal escape sequences shall be terminated to Catal and hexadecimal escape sequences shall be terminated to the terminated to Catal and hexadecimal escape sequences shall be terminated to the te	R.2.6	Advisory	declarations	610 S	Label is unused.
R.3.1 Required Used within a comment Used in // comments Used within a comment Used Used Within W	D 2 7	Advisory	There should be no unused parameters in	1 D	Unused procedure parameter.
R.3.1 Required Line-splicing shall not be used in // comments R.4.1 Required Line-splicing shall not be used in // comments R.4.2 Advisory Trigraphs should not be used 177 S Non standard escape sequence in source. R.5.1 Required External identifiers shall be distinct 177 D Identifier not unique within *** characters. Identifier match in *** characters. Identifier reuse: typedef vs macro. Identifier reuse: component vs macro. Identifier reuse: proc vs macro. Identifier reuse: proc vs enum constant. Identifier reuse: proc vs enum constant.	K.Z.1	Advisory	functions	15 D	Unused procedural parameter.
R.4.1 Required contained terminated terminat	R.3.1	Required		119 S	Nested comment found.
R.4.2 Advisory Trigraphs should not be used R.5.1 Required External identifiers shall be distinct R.5.2 Required Identifiers declared in the same scope and name space shall be distinct R.5.3 Required An identifier declared in an inner scope shall not hide an identifier declared in an outer scope R.5.4 Required Macro identifiers shall be distinct R.5.5 Required Identifiers shall be distinct R.5.6 Required Identifier and unique within *** characters. Identifier match in *** chars. Identifier not unique within *** characters. Identifier on tunique within *** characters. Identifier on tunique within *** characters. Identifier name reused. 92 S Upplicate use of a name in an enumeration. Parameter has same name as global variable. 131 S Name reused in inner scope. Identifier match in *** chars. Identifier match in *** chars. Identifier match in *** chars. Identifier match in **** chars. Identifier match in *** chars. Identifier reuse: tag vs macro. 21 X Identifier reuse: tag vs macro. 334 X Identifier reuse: tag vs macro. 34 X Identifier reuse: tag vs macro. 37 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: war vs macro. 57 X Identifier reuse: war ov s enum constant.	R.3.2	Required	Line-splicing shall not be used in // comments	611 S	Line splice used in // comment.
R.5.1 Required External identifiers shall be distinct R.5.2 Required Identifier declared in the same scope and name space shall be distinct R.5.3 Required Required Identifier declared in the same scope and name space shall be distinct R.5.3 Required An identifier declared in an inner scope shall not hide an identifier declared in an outer scope R.5.4 Required Macro identifiers shall be distinct R.5.5 Required Identifier out unique within *** characters. Identifier name reused. R.5.6 Required Identifier declared in an outer scope R.5.7 Required Identifier mach in *** characters. Identifier name reused. R.5.8 Required Identifier mach in macro name in an enumeration. R.5.9 Required Identifiers shall be distinct R.5.1 Required Identifier mach in macro name in 31 chars. R.5.2 Required Identifier shall be distinct from macro names R.5.3 Required Identifier reuse: proc vs macro. R.5.4 Required Identifier shall be distinct from macro names R.5.5 Required Identifier reuse: identifier reuse: proc vs macro. R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier reuse: identifier reuse: proc vs macro. R.5.5 Required Identifier reuse: identifier reuse: nacro vs macro. R.5.5 Required Identifier reuse: macro vs enum constant.	R.4.1	Required	· · · · · · · · · · · · · · · · · · ·	176 S	Non standard escape sequence in source.
R.5.1 Required External identifiers shall be distinct R.5.2 Required Identifier declared in the same scope and name space shall be distinct R.5.3 Required An identifier declared in an inner scope shall not hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier declared in an outer scope for independent of hide an identifier and the	R.4.2	Advisory	Trigraphs should not be used	81 S	Use of trigraph.
R.5.2 Required Identifiers declared in the same scope and name 17 D Identifier not unique within *** characters. 17 D Identifier not unique within *** characters. 18 D Identifier not unique within *** characters. 18 D Identifier name reused. 128 S Duplicate use of a name in an enumeration. 128 S Parameter has same name as global variable. 131 S Name reused in inner scope. 131 S Name reused in inner scope. 131 S Identifier match in *** characters. 142 S Identifier match in *** characters. 143 S Identifier name matches macro name in 31 chars. 144 S Identifier match in *** characters. 145 S S Identifier match in *** characters. 145 S S S Identifier match in *** characters. 145 S S S S Identifier match in *** characters. 145 S S S S S S S S S S S S S S S S S S S	D 5 1	Poquired	External identifiers shall be distinct	17 D	
R.5.3 Required space shall be distinct 61 X Identifier match in *** chars. An identifier declared in an inner scope shall not hide an identifier declared in an outer scope R.5.3 Required An identifier declared in an outer scope An identifier declared in an outer scope 128 S Duplicate use of a name in an enumeration. Parameter has same name as global variable. 128 S Parameter has same name as global variable. 131 S Name reused in inner scope. 61 X Identifier match in *** chars. 384 S Identifier matches macro name in 31 chars. 622 S Macro parameters are not unique within limits. 61 X Identifier name matches macro name in 31 chars. 61 X Identifier name matches macro name. 383 S Identifier reuse: tag vs macro. 21 X Identifier reuse: tag vs macro. 21 X Identifier reuse: proc vs macro. 31 X Identifier reuse: persistent var vs macro. 42 X Identifier reuse: persistent var vs macro. 43 X Identifier reuse: label vs macro. 44 X Identifier reuse: label vs macro. 45 X Identifier reuse: var vs macro. 55 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro se num constant.	K.5.1	Required	External identifiers shall be distinct	61 X	Identifier match in *** chars.
R.5.3 Required space shall be distinct 61 X Identifier match in *** chars. An identifier declared in an inner scope shall not hide an identifier declared in an outer scope R.5.3 Required An identifier declared in an outer scope An identifier declared in an outer scope 128 S Duplicate use of a name in an enumeration. Parameter has same name as global variable. 128 S Parameter has same name as global variable. 131 S Name reused in inner scope. 61 X Identifier match in *** chars. 384 S Identifier matches macro name in 31 chars. 622 S Macro parameters are not unique within limits. 61 X Identifier name matches macro name in 31 chars. 61 X Identifier name matches macro name. 383 S Identifier reuse: tag vs macro. 21 X Identifier reuse: tag vs macro. 21 X Identifier reuse: proc vs macro. 31 X Identifier reuse: persistent var vs macro. 42 X Identifier reuse: persistent var vs macro. 43 X Identifier reuse: label vs macro. 44 X Identifier reuse: label vs macro. 45 X Identifier reuse: var vs macro. 55 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro se num constant.	D.F.O	Doguirod	Identifiers declared in the same scope and name	17 D	Identifier not unique within *** characters.
R.5.3 Required An identifier declared in an inner scope shall not hide an identifier declared in an outer scope R.5.4 Required An identifier declared in an inner scope shall not hide an identifier declared in an outer scope R.5.4 Required Macro identifiers shall be distinct R.5.5 Required Macro identifiers shall be distinct R.5.5 Required Macro identifiers shall be distinct R.5.5 Required Macro identifiers shall be distinct Macro identifier macro name in 31 chars. An identifier match in *** chars. 12	K.5.2	Required	space shall be distinct	61 X	Identifier match in *** chars.
R.5.3 Required An identifier declared in an inner scope shall not hide an identifier declared in an outer scope R.5.4 Required Macro identifiers shall be distinct R.5.5 Required An identifier shall be distinct from macro names R.5.6 Required An identifier declared in an inner scope shall not hide an identifier declared in an outer scope 128 S Parameter has same name as global variable. 131 S Name reused in inner scope. 61 X Identifier match in *** chars. 384 S Identifier matches macro name in 31 chars. 622 S Imits. 61 X Identifier match in *** chars. 383 S Identifier name matches macro name. 384 S Identifier reuse: tag vs macro name. 384 S Identifier reuse: tag vs macro. 21 X Identifier reuse: typedef vs macro. 31 X Identifier reuse: proc vs macro. 32 X Identifier reuse: component vs macro. 43 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: proc param vs macro. 53 X Identifier reuse: proc param vs macro. 53 X Identifier reuse: macro vs enum constant.				17 D	Identifier not unique within *** characters.
R.5.3 Required An identifier declared in an inner scope shall not hide an identifier declared in an outer scope 128 S Parameter has same name as global variable. 131 S Name reused in inner scope. 61 X Identifier match in *** chars. 384 S Identifier matches macro name in 31 chars. 622 S Macro parameters are not unique within limits. 61 X Identifier match in *** chars. 383 S Identifier match in *** chars. 384 S Identifier match in *** chars. 383 S Identifier match in *** chars. 384 S Identifier match in *** chars. 385 S Identifier match in *** chars. 386 S Identifier reuse: proc variance in 31 chars. 12 X Identifier reuse: tag vs macro. 384 X Identifier reuse: proc vs macro. 385 S Identifier reuse: proc vs macro. 48 X Identifier reuse: component vs macro. 48 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: var vs macro. 53 X Identifier reuse: macro vs enum constant.				18 D	
R.5.4 Required Macro identifiers shall be distinct R.5.4 Required Macro identifiers shall be distinct R.5.5 Required Identifiers shall be distinct from macro names R.5.5 Required Identifiers shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier reuse: tag vs macro. R.5.6 Required Identifier reuse: tag vs macro. R.5.7 X Identifier reuse: component vs macro. AND Identifier reuse: label vs macro (MR). AND Identifier reuse: proc param vs macro. 53 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro vs enum constant.	D 5 3	Required		92 S	Duplicate use of a name in an enumeration.
R.5.4 Required Macro identifiers shall be distinct Required Macro identifiers shall be distinct R.5.4 Required Macro identifiers shall be distinct R.5.5 Required Identifiers shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier reuse: tag vs macro. 21 X Identifier reuse: typedef vs macro. 34 X Identifier reuse: proc vs macro. 47 X Identifier reuse: component vs macro. 48 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: var vs macro. 53 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro vs enum constant.	N.3.3				variable.
R.5.4 Required Macro identifiers shall be distinct Required Macro identifier shall be distinct in *** chars. Required Macro parameters are not unique within limits. Required Macro parameters are no					Name reused in inner scope.
R.5.4 Required Macro identifiers shall be distinct 622 S Macro parameters are not unique within limits. 61 X Identifier match in *** chars. 383 S Identifier name matches macro name. 384 S Identifier matches macro name in 31 chars. 12 X Identifier reuse: tag vs macro. 21 X Identifier reuse: typedef vs macro. 34 X Identifier reuse: proc vs macro. 37 X Identifier reuse: persistent var vs macro. 47 X Identifier reuse: component vs macro. 48 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: proc param vs macro. 53 X Identifier reuse: macro vs enum constant.				61 X	Identifier match in *** chars.
R.5.5 Required Identifiers shall be distinct from macro names R.5.5 Required Requ			Required Macro identifiers shall be distinct	384 S	Identifier matches macro name in 31 chars.
R.5.5 Required Identifiers shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier reuse: tag vs macro. 21 X Identifier reuse: typedef vs macro. 34 X Identifier reuse: proc vs macro. 37 X Identifier reuse: persistent var vs macro. 47 X Identifier reuse: component vs macro. 48 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: var vs macro. 53 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro vs enum constant.	R.5.4	Required		622 S	limits.
R.5.5 Required Identifiers shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names R.5.5 Required Identifier reuse: typedef vs macro. 34 X Identifier reuse: proc vs macro. 37 X Identifier reuse: persistent var vs macro. 47 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: var vs macro. 53 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro vs enum constant.					
R.5.5 Required				383 S	Identifier name matches macro name.
R.5.5 Required Identifiers shall be distinct from macro names R.5.5 Required Identifier shall be distinct from macro names 21 X Identifier reuse: typedef vs macro.					
R.5.5 Required Identifiers shall be distinct from macro names A X Identifier reuse: proc vs macro.					
R.5.5 Required Identifiers shall be distinct from macro names 37 X Identifier reuse: persistent var vs macro. 47 X Identifier reuse: component vs macro. 48 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: var vs macro. 53 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro vs enum constant.				21 X	Identifier reuse: typedef vs macro.
47 X Identifier reuse: component vs macro. 48 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: var vs macro. 53 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro vs enum constant.				34 X	Identifier reuse: proc vs macro.
48 X Identifier reuse: label vs macro (MR). 50 X Identifier reuse: var vs macro. 53 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro vs enum constant.	R.5.5	Required	Identifiers shall be distinct from macro names	37 X	Identifier reuse: persistent var vs macro.
50 X Identifier reuse: var vs macro. 53 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro vs enum constant.				47 X	Identifier reuse: component vs macro.
53 X Identifier reuse: proc param vs macro. 57 X Identifier reuse: macro vs enum constant.				48 X	Identifier reuse: label vs macro (MR).
57 X Identifier reuse: macro vs enum constant.				50 X	Identifier reuse: var vs macro.
57 X Identifier reuse: macro vs enum constant.					

			112 S	Typedef name redeclared.
			374 S	Name conflict with typedef.
			11 X	Identifier reuse: tag vs typedef.
			16 X	Identifier reuse: typedef vs variable.
			17 X	Identifier reuse: typedef vs label (MR).
R.5.6	Doguirod	A typedef name shall be a unique identifier	18 X	Identifier reuse: typedef vs typedef.
K.5.0	Required	A typedef name shall be a unique identifier	19 X	Identifier reuse: typedef vs procedure
			19 X	parameter.
			20 X	Identifier reuse: persistent var vs typedef.
			22 X	Identifier reuse: typedef vs component.
			23 X	Identifier reuse: typedef vs enum constant.
			24 X	Identifier reuse: typedef vs procedure.
			325 S	Inconsistent use of tag.
			4 X	Identifier reuse: struct/union tag repeated.
			5 X	Identifier reuse: struct vs union.
			6 X	Identifier reuse: struct/union tag vs enum
			0 1	tag.
			7 X	Identifier reuse: tag vs procedure.
R.5.7	Required	A tag name shall be a unique identifier	8 X	Identifier reuse: tag vs procedure parameter.
			9 X	Identifier reuse: tag vs variable.
			10 X	Identifier reuse: tag vs label (MR).
			11 X	
			13 X	Identifier reuse: tag vs component.
				Identifier reuse: tag vs enum constant.
				Identifier reuse: persistent var vs tag.

			1 S	Procedure name reused.
			7 X	Identifier reuse: tag vs procedure.
			15 X	Identifier reuse: persistent var vs tag.
			20 X	Identifier reuse: persistent var vs typedef.
			24 X	Identifier reuse: typedef vs procedure.
			25 X	Identifier reuse: procedure vs procedure param.
			26 X	Identifier reuse: persistent var vs label (MR).
			27 X	Identifier reuse: persist var vs persist var.
		Identifiers that define objects or functions with external linkage shall be unique	28 X	Identifier reuse: persistent var vs var.
R.5.8	Required		29 X	Identifier reuse: persistent var vs procedure.
			30 X	Identifier reuse: persistent var vs proc param.
			31 X	Identifier reuse: procedure vs procedure.
			32 X	Identifier reuse: procedure vs var.
			33 X	Identifier reuse: procedure vs label (MR).
			35 X	Identifier reuse: proc vs component.
			36 X	Identifier reuse: proc vs enum constant.
			38 X	Identifier reuse: persistent var vs
			30 X	component.
			39 X	Identifier reuse: persistent var vs enum
			33 X	constant.

				<u> </u>
			1 S	Procedure name reused.
			7 X	Identifier reuse: tag vs procedure.
				Identifier reuse: persistent var vs tag.
				Identifier reuse: persistent var vs typedef.
			24 X	Identifier reuse: typedef vs procedure.
			25 X	Identifier reuse: procedure vs procedure
			25 X	param.
			26 X	Identifier reuse: persistent var vs label (MR).
			27 X	Identifier reuse: persist var vs persist var.
			28 X	Identifier reuse: persistent var vs var.
R.5.9	Advisory	Identifiers that define objects or functions with internal linkage should be unique	29 X	Identifier reuse: persistent var vs procedure.
			30 X	Identifier reuse: persistent var vs proc
				param.
			31 X	Identifier reuse: procedure vs procedure.
				Identifier reuse: procedure vs var.
				Identifier reuse: procedure vs label (MR).
				Identifier reuse: proc vs component.
			36 X	Identifier reuse: proc vs enum constant.
			38 X	Identifier reuse: persistent var vs
			00 X	component.
			39 X	Identifier reuse: persistent var vs enum
				constant.
R.6.1	Required	Bit-fields shall only be declared with an appropriate	73 S	Bit field not signed or unsigned int.
	7.10 4	type	520 S	Bit field is not bool or explicit integral.
R.6.2	Required	Single-bit named bit fields shall not be of a signed type	72 S	Signed bit field less than 2 bits wide.
R.7.1	Required	Octal constants shall not be used	83 S	Octal number found.
R.7.2	Required	A "u" or "U" suffix shall be applied to all integer	331 S	Literal value requires a U suffix.
13.7.2	rtoquilou	constants that are represented in an unsigned type	550 S	Unsuffixed hex or octal is unsigned, add U.
R.7.3	Required	The lowercase character 'l' shall not be used in a literal suffix	252 S	Lower case suffix to literal number.
R.7.4	Required	A string literal shall not be assigned to an object	157 S	Modification of string literal.
11.7.7	Required	unless the object's type is "pointer to const-	623 S	String assigned to non const object.
_			20 S	Parameter not declared explicitly.
R.8.1	Required	Types shall be explicitly specified	135 S	Parameter list is KR.
			326 S	Declaration is missing type.

			•	
		Function types shall be in prototype form with	37 S	Procedure parameter has a type but no identifier.
R.8.2	Required	named parameters	63 S	Empty parameter list to procedure/function.
			135 S	Parameter list is KR.
		All declarations of an abject on function about the	36 D	Prototype and definition name mismatch.
R.8.3	Required	All declarations of an object or function shall use the same names and type qualifiers	63 X	Function prototype/defn param type mismatch (MR).
			36 D	Prototype and definition name mismatch.
			106 D	No prototype for non-static function.
			102 S	Function and prototype return inconsistent (MR).
R.8.4	Required	A compatible declaration shall be visible when an	103 S	Function and prototype param inconsistent (MR).
11.0.4	rrequired	object or function with external linkage is defined	1 X	Declaration types do not match across a system.
			62 X	Function prototype/defn return type mismatch (MR).
			63 X	Function prototype/defn param type mismatch (MR).
R.8.5	Required	An external object or function shall be declared	60 D	External object should be declared only once.
K.0.5	Required	once in one and only one file	110 D	More than one prototype for same function.
			172 S	Variable declared multiply.
			26 D	Variable should be defined once in only one file.
R.8.6	Required	An identifier with external linkage shall have	33 D	No real declaration for external variable.
11.0.0	rtoquirou	exactly one external definition	34 D	Procedure name re-used in different files.
			63 D	No definition in system for prototyped procedure.
R.8.7	Advisory	Functions and objects should not be defined with	27 D	Variable should be declared static.
11.0.7	7 (0 1 1 0 0 1)	external linkage if they are referenced in only one	61 D	Procedure should be declared static.
			27 D	Variable should be declared static.
D 0 0	D	The static storage class specifier shall be used in	61 D	Procedure should be declared static.
R.8.8	Required	all declarations of objects and functions that have	461 S	Identifier with ambiguous linkage.
		internal linkage	553 S	Function and proto should both be static.
		An object abould be defined at black access if its	575 S	Linkage differs from previous declaration.
R.8.9	Advisory	An object should be defined at block scope if its identifier only appears in a single function	25 D	Scope of variable could be reduced.

			1	
R.8.10	Required	An inline function shall be declared with the static storage class	612 S	inline function should be declared static.
R.8.11	Advisory	When an array with external linkage is declared, its size should be explicitly specified	127 S	Array has no bounds specified.
R.8.12	Required	Within an enumerator list, the value of an implicitly- specified enumeration constant shall be unique	630 S	Duplicated enumeration value.
R.8.13	Advisory	A pointer should point to a const-qualified type whenever possible	120 D	Pointer param should be declared pointer to const.
R.8.14	Required	The restrict type qualifier shall not be used	613 S	Use of restrict keyword.
			53 D	Attempt to use uninitialised pointer.
R.9.1	Mandatory	The value of an object with automatic storage duration shall not be read before it has been set	69 D	UR anomaly, variable used before assignment.
N.9.1			631 S	Declaration not reachable.
			652 S	Object created by malloc used before initialisation.
R.9.2	Required	The initializer for an aggregate or union shall be enclosed in braces	105 S	Initialisation brace { } fault.
R.9.3	Required	Arrays shall not be partially initialized	397 S	Array initialisation has insufficient items.
R.9.4	Required	An element of an object shall not be initialised	620 S	Initialisation designator duplicated.
11.5.4	rrequired	more than once	627 S	Initialiser both positional and designational.
R.9.5	Required	Where designated initialisers are used to initialize an array object the size of the array shall be specified explicitly	127 S	Array has no bounds specified.

			50 S	Use of shift operator on signed type.
			52 S	Unsigned expression negated.
			93 S	Value is not of appropriate type.
			96 S	Use of mixed mode arithmetic.
			109 S	Array subscript is not integral.
			114 S	Expression is not Boolean.
			120 S	Use of bit operator on signed type.
R.10.1	Required	Operands shall not be of an inappropriate essential type	123 S	Use of underlying enum representation value.
		essential type	136 S	Bit operator with boolean operand.
			249 S	Operation not appropriate to boolean type.
			329 S	Operation not appropriate to plain char.
			345 S	Bit operator with floating point operand.
			389 S	Bool value incremented/decremented.
			403 S	Negative (or potentially negative) shift.
			433 S	Type conversion without cast.
			506 S	Use of boolean with relational operator.
R.10.2	Required	Expressions of essentially character type shall not	96 S	Use of mixed mode arithmetic.
11.10.2	Required	be used inappropriately in addition and subtraction	329 S	Operation not appropriate to plain char.
			93 S	Value is not of appropriate type.
			96 S	Use of mixed mode arithmetic.
			101 S	Function return type inconsistent.
			104 S	Struct field initialisation incorrect.
			123 S	Use of underlying enum representation value.
			276 S	Case is not part of switch enumeration.
			330 S	Implicit conversion of underlying type (MR).
		The value of an expression shall not be assigned	331 S	Literal value requires a U suffix.
R.10.3	Required	to an object with a narrower essential type or of a	411 S	Inappropriate value assigned to enum.
11.10.5	Required	different essential type category	431 S	Char used instead of (un)signed char.
		different essential type category	432 S	Inappropriate type - should be plain char.
			433 S	Type conversion without cast.
			434 S	Signed/unsigned conversion without cast.
			435 S	Float/integer conversion without cast.
			445 S	Narrower float conversion without cast.
			446 S	Narrower int conversion without cast.
			458 S	Implicit conversion: actual to formal param (MR).
			488 S	Value outside range of underlying type.

			93 S	Value is not of appropriate type.
			96 S	Use of mixed mode arithmetic.
			107 S	Type mismatch in ternary expression.
			123 S	Use of underlying enum representation
		Both operands of an operator in which the usual	123 3	value.
R.10.4	Required	arithmetic conversions are performed shall have	330 S	Implicit conversion of underlying type (MR).
		the same essential type category	331 S	Literal value requires a U suffix.
			433 S	Type conversion without cast.
			434 S	Signed/unsigned conversion without cast.
			435 S	Float/integer conversion without cast.
			488 S	Value outside range of underlying type.
R.10.5	Advisory	The value of an expression should not be cast to	93 S	Value is not of appropriate type.
K. 10.5	Advisory	an inappropriate essential type	95	value is flot of appropriate type.
			451 S	No cast for widening complex float
R.10.6	Required	The value of a composite expression shall not be	451 0	expression (MR).
K. 10.0	Required	assigned to an object with wider essential type	452 S	No cast for widening complex int expression
			452 5	(MR).
	Required	If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type	451 S	No cast for widening complex float
R.10.7			451 5	expression (MR).
13.10.7			452 S	No cast for widening complex int expression
			452 5	(MR).
	Required		332 S	Widening cast on complex integer
				expression (MR).
		The value of a composite expression shall not be cast to a different essential type category or a wider essential type	333 S	Widening cast on complex float expression
R.10.8				(MR).
11.10.0			441 S	Float cast to non-float.
		wider essertial type	442 S	Signed integral type cast to unsigned.
			443 S	Unsigned integral type cast to signed.
			444 S	Integral type cast to non-integral.
R.11.1	Required		93 S	Value is not of appropriate type.
		Conversions shall not be performed between a	94 S	Casting operation on a pointer.
		pointer to a function and any other type	95 S	Casting operation to a pointer.
		pointer to a function and any other type	440 S	Cast from integral type to pointer.
			606 S	Cast involving function pointer.
			94 S	Casting operation on a pointer.
	Required	Conversions shall not be performed between a pointer to incomplete and any other type	95 S	Casting operation to a pointer.
R.11.2			439 S	Cast from pointer to integral type.
			440 S	Cast from integral type to pointer.
			554 S	Cast to an unrelated type.

				1
5		A cast shall not be performed between a pointer to	94 S	Casting operation on a pointer.
R.11.3	Required	object type and a pointer to a different object type	95 S	Casting operation to a pointer.
		, ,,	554 S	Cast to an unrelated type.
R.11.4	Advisory	A conversion should not be performed between a	439 S	Cast from pointer to integral type.
13.111.4	7 (0 1) 0 1 9	pointer to object and an integer type	440 S	Cast from integral type to pointer.
R.11.5	Advisory	A conversion should not be performed from pointer	95 S	Casting operation to a pointer.
14.11.0	Advisory	to void into pointer to object	433 S	Type conversion without cast.
			439 S	Cast from pointer to integral type.
R.11.6	Required	A cast shall not be performed between pointer to	440 S	Cast from integral type to pointer.
13.11.0	Required	void and an arithmetic type	635 S	Cast from pointer to float type.
			636 S	Cast from float type to pointer.
			94 S	Casting operation on a pointer.
			95 S	Casting operation to a pointer.
R.11.7	Doguirod	A cast shall not be performed between pointer to	439 S	Cast from pointer to integral type.
K.11.7	Required	object and a non-integer arithmetic type	440 S	Cast from integral type to pointer.
			635 S	Cast from pointer to float type.
			636 S	Cast from float type to pointer.
D 44.0	5	A cast shall not remove any const or volatile	203 S	Cast on a constant value.
R.11.8	Required	qualification from the type pointed to by a pointer	344 S	Cast on volatile value.
R.11.9	Required	Required The macro NULL shall be the only permitted form of integer null pointer constant	531 S	Literal zero used in pointer context.
17.11.9			3313	Literal zero used in pointer context.
R.12.1	Advisory	Advisory The precedence of operators within expressions	49 S	Logical conjunctions need brackets.
11.12.1		should be made explicit	361 S	Expression needs brackets.
R.12.2	Required	The right hand operand of a shift operator shall lie	51 S	Shifting value too far.
11.12.2	Required	in the range zero to one less than the width in bits	403 S	Negative (or potentially negative) shift.
R.12.3	Advisory	The comma operator should not be used	53 S	Use of comma operator.
R.12.4	Advisory	Evaluation of constant expressions should not lead	493 S	Numeric overflow.
11.12.4	Advisory	to unsigned integer wrap-around	494 S	Numeric underflow.
	Mandatory	The sizeof operator shall not have an operand	401 S	
R.12.5		which is a function parameter declared as "array of		Use of sizeof on an array parameter.
		type"	05 D	
R.13.1	r Required I		35 D	Expression has side effects.
			1 Q	Call has execution order dependant side
				effects.
		Initialiser lists shall not contain persistent side	9 S	Assignment operation in expression.
		red effects	30 S	Deprecated usage of ++ or operators
				found.
			132 S	Assignment operator in boolean expression.
			134 S	Volatile variable in complex expression.
			1343	volatile variable in complex expression.

R.13.2 Required				35 D	Expression has side effects.
R.13.2 Required Requi					
R.13.3 Advisory A full expression containing an increment (++) or decrement () operator should have no other potential side effects other than that caused by the increment or decrement operator should not be used R.13.4 Advisory R.13.5 Required Required R.13.6 Mandatory R.13.6 Required		Descriped			Call has execution order dependant side
R.13.3 Advisory A full expression containing an increment (++) or decrement () operator should have no other potential side effects other than that caused by the increment or decrement operator R.13.4 Advisory R.13.5 Required Required Required A full expression containing an increment (++) or decrement () operator should have no other potential side effects other than that caused by the increment or decrement operator The result of an assignment operator should not be used The result of an assignment operator should not be used The right hand operand of a logical && or operator shall not contain persistent side effects The right hand operand of a logical && or operator shall not contain persistent side effects The operand of the size of operator shall not contain any expression which has potential side effects R.14.1 Required A full expression containing an increment (++) or decrement (++) or decrement (+-) operator should have no other potential side effects of the found. B A ssignment operation in expression. A ssignment operation in expression. A ssignment operation in boolean expression. Call has execution order dependant side effects. Use of ++ or on RHS of && or operator on the contain any expression which has potential side effects. Size of operator with side effects. Size of operator with side effects. Unsuitable type for loop variable.	R.13.2	Requirea	· · · · · · · · · · · · · · · · · · ·	9 S	Assignment operation in expression.
R.13.3 Advisory A full expression containing an increment (++) or decrement () operator should have no other potential side effects other than that caused by the increment or decrement operator R.13.4 Advisory The result of an assignment operator should not be used The right hand operand of a logical && or operator shall not contain persistent side effects The operand of the sizeof operator shall not contain any expression which has potential side effects R.13.6 Required A full expression containing an increment (++) or decrement (++) or decrement () operator should have no other potential side effects assignment operator in expression. 9 S Assignment operation in expression. 132 S Assignment operator in boolean expression Call has execution order dependant side effects. Use of ++ or on RHS of && or operator with side effects. The operand of the sizeof operator shall not contain any expression which has potential side effects. R.14.1 Required A loop counter shall not have essentially floating type A loop counter in loop body.			evaluation orders	30 S	
R.13.3 Advisory decrement () operator should have no other potential side effects other than that caused by the increment or decrement operator R.13.4 Advisory The result of an assignment operator should not be used R.13.5 Required The right hand operand of a logical && or operator shall not contain persistent side effects R.13.6 Mandatory The operand of the sizeof operator shall not contain any expression which has potential side effects R.14.1 Required A loop counter shall not have essentially floating type Advisory Deprecated usage of ++ or operators found. 30 S Deprecated usage of ++ or operators found. 9 S Assignment operation in expression. 132 S Assignment operator in boolean expression. 140 Call has execution order dependant side effects. 10 Call has execution order dependant side effects. 406 S Use of ++ or on RHS of && or operator. 10 Call has execution order dependant side effects. 10 Call has execution or effects. 10 Call has execution order dependant side effects. 10 Call has execution order dependant side effects. 10 Call has execution order dependant side effects. 10 Call has execution or e				134 S	Volatile variable in complex expression.
R.13.4 Advisory be used 132 S Assignment operator in boolean expression be used 132 S Assignment operator in boolean expression be used 132 S Assignment operator in boolean expression be used 132 S Assignment operator in boolean expression be used 132 S Assignment operator in boolean expression be used 132 S Assignment operator in boolean expression be used 132 S D Expression has side effects. 1 Q Call has execution order dependant side effects. 406 S Use of ++ or on RHS of && or operator shall not contain any expression which has potential side effects. 1 Q Call has execution order dependant side effects. 2 Volatile variable accessed on RHS of && or operator shall not contain any expression which has potential side effects. 2 Sizeof operator with side effects. 3 Sizeof operator with side effects. 3 Sizeof operator with side effects. 3 Sizeof operator with side effects. 4 I O D D Sizeof operator with side effects. 5 Sizeof operator with side effects.	R.13.3	Advisory	decrement () operator should have no other potential side effects other than that caused by the	30 S	·
R.13.5 Required Required			The result of an assignment operator should not	9 S	Assignment operation in expression.
R.13.5 Required The right hand operand of a logical && or operator shall not contain persistent side effects The right hand operand of a logical && or operator shall not contain persistent side effects. 406 S Use of ++ or on RHS of && or operator shall not contain any expression which has potential side effects. R.13.6 Required The operand of the sizeof operator shall not contain any expression which has potential side effects A loop counter shall not have essentially floating type A loop counter shall not have essentially floating type Unsuitable type for loop variable.	R.13.4	Advisory	· ·	132 S	Assignment operator in boolean expression.
R.13.5 Required The right hand operand of a logical && or operator shall not contain persistent side effects The operator shall not contain persistent side effects 406 S Use of ++ or on RHS of && or operator shall variable accessed on RHS of && or operator shall not contain any expression which has potential side effects R.13.6 Required Required A loop counter shall not have essentially floating type 39 S Unsuitable type for loop variable. 55 D Modification of loop counter in loop body.				35 D	Expression has side effects.
Parameter of the size of the size of operator shall not contain persistent side effects Align: Contain any expression which has potential side effects 408 S Use of ++ or on RHS of && or operator	D 12 5	Required		1 Q	·
R.13.6 Mandatory The operand of the sizeof operator shall not contain any expression which has potential side effects R.14.1 Required A loop counter shall not have essentially floating type Sizeof operator with side effects. Unsuitable type for loop variable. 55 D Modification of loop counter in loop body.	N. 13.3			406 S	Use of ++ or on RHS of && or operator.
R.13.6 Mandatory contain any expression which has potential side effects. R.14.1 Required A loop counter shall not have essentially floating type Size of operator with side effects. Unsuitable type for loop variable. 55 D Modification of loop counter in loop body.				408 S	Volatile variable accessed on RHS of && or .
type to loop variable. 1	R.13.6	Mandatory	contain any expression which has potential side	54 S	Sizeof operator with side effects.
	R.14.1	Required	,	39 S	Unsuitable type for loop variable.
270 S For loop initialisation is not simple.					
R 14.7 Reduired TA for loop shall be Well-formed	R 14 2	Required	A for loop shall be well-formed		
429 S Empty middle expression in for loop.	N. 14.2	rtoquilou	A to loop onal so won formed		
					Inconsistent usage of loop control variable.
581 S Loop conditions are independent.					
R.14.3 Required Controlling expressions shall not be invariant 139 S Construct leads to infeasible code.	R.14.3	Required	Controlling expressions shall not be invariant		
140 S Infeasible loop condition found.		•	ı ı	140 S	inreasible loop condition found.
R.14.4 Required The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type The controlling expression of an if statement and the controlling expression of an iteration-statement and shall have essentially Boolean type	R.14.4	Required	the controlling expression of an iteration-statement	114 S	Expression is not Boolean.
R.15.1 Advisory The goto statement should not be used 13 S goto detected.	R.15.1	Advisory	The goto statement should not be used	13 S	goto detected.

R.15.2	Required	The goto statement shall jump to a label declared later in the same function	509 S	goto label is backwards.	
R.15.3	Required	Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement	511 S	Jump into nested block.	
R.15.4	Advisory	There should be no more than one break or goto statement used to terminate any iteration statement	409 S	More than one break or goto statement in loop.	
R.15.5	Advisory	A function should have a single point of exit at the end	7 C	Procedure has more than one exit point.	
		The body of an iteration-statement or a selection-	11 S	No brackets to loop body (added by Testbed).	
R.15.6	Required	statement shall be a compound statement	12 S	No brackets to then/else (added by Testbed).	
			428 S	No {} for switch (added by Testbed).	
D 45.7	Danishad	All if else if constructs shall be terminated with	59 S	Else alternative missing in if.	
R.15.7	Required	an else statement	477 S	Empty else clause following else if.	
R.16.1	Required	All switch statements shall be well-formed	385 S	MISRA switch statement syntax violation.	
R.16.2	Required	A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	245 S	Case statement in nested block.	
R.16.3	Required	An unconditional break statement shall terminate every switch-clause	62 S	Switch case not terminated with break.	
			48 S	No default case in switch statement.	
R.16.4	Required	Every switch statement shall have a default label	410 S	Switch empty default has no comment (MR).	
R.16.5	Required	A default label shall appear as either the first or the last switch label of a switch statement	322 S	Default is not last case of switch.	
R.16.6	Required	Every switch statement shall have at least tw	Every switch statement shall have at least two	60 S	Empty switch statement.
K. 10.0		switch-clauses	61 S	Switch contains default only.	
R.16.7	Required	A switch-expression shall not have essentially Boolean type	121 S	Use of boolean expression in switch.	
R.17.1	Required	The features of <stdarg.h> shall not be used</stdarg.h>	44 S	Use of banned function, type or variable.	
D 47.0		Functions shall not call themselves, either directly	6 D	Recursion in procedure calls found.	
R.17.2	Required	or indirectly		Inter-file recursion found.	
R.17.3	Mandatory	A function shall not be declared implicitly	496 S	Function call with no prior declaration.	

R.17.4 Mandatory All exit paths from a function with non-void return type shall have an explicit return statement with an expression R.17.5 Advisory The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements R.17.6 Mandatory R.17.7 Required R.17.7 Required R.17.8 Advisory A function parameter should not be modified R.18.1 Required A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand shall address an element of the same array be applied to pointers that address elements of the same array R.18.2 Required R.18.3 Required R.18.4 Advisory A dvisory A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand R.18.3 Required R.18.4 Advisory A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand R.18.3 Required R.18.4 Advisory A pointer resulting from arithmetic on a pointer operand shall address an element of the same array bear and exceeded. A pointer resulting from arithmetic on a pointer operand shall address an element of the same array bear and exceeded. A pointer resulting from arithmetic on a pointer operand shall address an element of the same array bear and exceeded at call. A pointer resulting from arithmetic on a pointer operand shall address an element of the same array bear and exceeded at call. A pointer arithmetic is not on array. B pointer arithmetic is not on array. B pointer arithmetic is not on array. B pointer subtraction not addressing one array. A pointer subtraction not addressing one array. The relational operators > > A constant parameter indexing array too small at call. B pointer arithmetic is not on array. A pointer assignment to wind array bear array bear and a pointer type applied to objects of pointer rype except where they point into the same object. A pointer assignment to winder s					
R.17.5 Advisory The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements R.17.6 Mandatory The declaration of an array parameter shall not contain the static keyword between the [1] through the static keyword t	D 47.4	Mandatan		2 D	Function does not return a value on all paths.
R.17.5 Advisory R.17.6 Mandatory R.17.7 Required R.17.8 Advisory R.17.8 Advisory R.17.8 Required R.17.8 Advisory R.17.8 Required R.17.8 Advisory R.17.8 Advisory R.17.8 Required R.18.1 Required R.18.1 Required R.18.1 Required R.18.2 Required R.18.2 Required R.18.3 Required R.18.3 Required R.18.4 Advisory R.18.5 Advisory R.18.5 Advisory R.18.6 Required R.18.6 Required R.18.6 Required R.18.6 Required R.18.7 Required R.18.7 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.9 Required R.18.1 Required R.18.1 Required R.18.2 Required R.18.3 Required R.18.4 Required R.18.5 Required R.18.6 Required R.18.6 Required R.18.7 Required R.18.7 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.9 Required R.18.0 Required R.18.1 Required R.18.1 Required R.18.2 Required R.18.3 Required R.18.4 Required R.18.5 Required R.18.6 Required R.18.6 Required R.18.7 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.9 Required R.18.0 Required R.18.1 Required R.18.1 Required R.18.2 Required R.18.3 Required R.18.4 Advisory R.18.5 Advisory R.18.6 Required R.18.7 Required R.18.7 Required R.18.8 Required R.18.8 Required R.18.9 Required R.18.0 Required R.18.0 Required R.18.1 Required R.18.1 Required R.18.2 Required R.18.3 Required R.18.4 Advisory R.18.4 Advisory R.18.5 Advisory R.18.6 Required R.18.7 Required R.18.7 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.9 Required R.18.1 Required R.18.1 Required R.18.2 Required R.18.3 Required R.18.4 Required R.18.4 Required R.18.5 Required R.18.6 Required R.18.6 Required R.18.7 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.9 Required R.18.9 Required R.18.1 Required R.18.1 Required R.18.2 Required R.18.3 Required R.18.4 Required R.18.4 Required R.18.5 Required R.18.6 Required R.18.7 Required R.18.8 Required R.18.8 Required R.18.9 Required R.18.9 Required R.18.9 Required R.18.	R.17.4	iviandatory	• • • • • • • • • • • • • • • • • • • •	36 S	Function has no return statement.
R.17.5 Advisory The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements			expression		
R.17.6 Mandatory R.17.7 Required Required Required Required Required R.17.8 Advisory A function parameter should not be modified R.17.8 Advisory A function parameter should not be modified R.18.1 Required A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand R.18.2 Required R.18.3 Required R.18.4 Advisory R.18.5 Advisory Contain the static keyword between the [] The value returned by a function having non-void return value potentially unused. 332 S (void) missing for discarded return value. 41 D Attempt to change parameter passed by value. 42 S Array bound exceeded. 436 S Declaration does not specify an array. 437 S Array bound exceeded at call. 438 S Parameter indexing array too big at call. 638 X Parameter indexing array too big at call. 638 X Parameter indexing array too small at call. 639 S Alray index is negative. 640 X Array bound exceeded at call. 657 S Pointer arithmetic is not on array. 658 S Pointer arithmetic is not on array. 659 S Pointer subtraction not addressing one array. 72 X Parameter indexing array too small at call. 650 S Pointer subtraction not addressing one array. 72 X Parameter indexing array too small at call. 650 S Pointer subtraction not addressing one array. 72 X Parameter indexing array too small at call. 650 S Pointer subtraction not addressing one array. 750 Pointer subtraction not addressing one array. 750 Pointer arithmetic is not on array. 750 Pointer assignment to wider scope. 750 Pointer assignment to wider scope. 750 Pointe	R.17.5	Advisory	parameter declared to have an array type shall		
R.17.8 Advisory A function parameter should not be modified R.17.8 Advisory A function parameter should not be modified R.18.1 Required A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand R.18.2 Required Subtraction between pointers shall only be applied to pointers that address elements of the same array R.18.3 Required The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object R.18.4 Advisory The +, += and -= operators should not be applied to an expression of pointer type R.18.5 Advisory Declarations should contain no more than two levels of pointer nesting R.18.6 Required Flexible array members shall not be declared R.18.7 Required Flexible array members shall not be declared R.18.8 Required The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist R.18.7 Required Flexible array members shall not be declared R.18.8 Required The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist R.18.7 Required Flexible array members shall not be declared R.18.8 Required The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist R.18.9 Required Flexible array members shall not be declared R.18.9 Required Flexible array members shall not be declared R.18.9 Required Flexible array members shall not be declared R.18.9 Reguired Flexible array members shall not be declared R.18.9 Reguired Flexible array members shall not be declared R.18.1 Required Flexible array members shall not be declared R.18.1 Required Flexible array members shall not be declared R.18.1 Required Flexible array members shall not be declared R.18.1 Required Flexible array members shall not be declared R.18.1 Required Flexible array members	R.17.6	Mandatory		614 S	Use of static keyword in array parameter.
R.17.8 Advisory A function parameter should not be modified R.17.8 Advisory A function parameter should not be modified R.18.1 Required A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand shall address an element of the same array as that pointer operand R.18.2 Required Subtraction between pointers shall only be applied to pointers that address elements of the same array R.18.3 Required The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object R.18.4 Advisory The +, += and == operators should not be applied to an expression of pointer type R.18.5 Advisory Declarations should contain no more than two levels of pointer resting R.18.6 Required Required Flexible array members shall not be declared Agray members shall not be declared Agray bound exceeded at use. 72 X Parameter indexing array too big at call. 69 X Global array bound exceeded at use. 72 X Parameter indexing array too small at call. 80 Y Solice array bound exceeded at use. 80 X Global array bound exceeded at use. 80 X Pointer subtraction not addressing one array. 80 Y Solice array bound exceeded at call. 80 X Pointer subtraction not addressing one array. 80 Y Solice array bound exceeded at call. 80 Y Pointer arithmetic solice array. 80 Y Solice array bound exceeded at call. 80 Y Solice array bound exceeded at call. 80 Y Pointer arithmetic solice array. 80 Y Solice array bound exceeded at call. 80 Y Pointer arithmetic solice array. 81 Y S Y S Y Solice array bound exceeded at call. 82 Y S Y Parameter indexing array too big at call. 83 Y Pointer arithmetic solice array. 84 Y S Y S Y S Y S Y S Y S Y S Y S Y S Y	D 17 7	Doguirod	The value returned by a function having non-void	91 D	Function return value potentially unused.
R.17.8 Advisory A function parameter should not be modified R.18.1 Required A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand R.18.2 Required Subtraction between pointers shall only be applied to pointers that address elements of the same array R.18.3 Required Subtraction between pointer shall only be applied to pointers that address elements of the same array R.18.4 Advisory The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object R.18.5 Advisory Declarations should contain no more than two levels of pointer nesting The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist Advisory Required Flexible array members shall not be declared 4815 Array with no bounds in struct.	K.17.7	Required	return type shall be used	382 S	(void) missing for discarded return value.
R.18.1 Required A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand R.18.2 Required Subtraction between pointers shall only be applied to pointers that address elements of the same array R.18.3 Required The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object R.18.4 Advisory R.18.5 Advisory R.18.6 Required Required Required R.18.6 Required Required R.18.6 Required Required R.18.7 Required R.18.7 Required R.18.7 Required R.18.7 Required Required R.18.7 Required Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.8 Required Required Required R.18.8 Required R.18.8 Required Required R.18.8 Required Required R.18.8 Required Required Required R.18.8 Required Required R.18.8 Required Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.8 Required Required Required R.18.8 Required Require	R.17.8	Advisory			Attempt to change parameter passed by value.
R.18.1 Required R.18.2 Required R.18.3 Required R.18.3 Required R.18.4 Required R.18.4 Required R.18.5 Required R.18.5 Required R.18.6 Required R.18.6 Required R.18.6 Required R.18.7 Required Required Required R.18.7 Required Required Required Required Required R.18.8 Required R.18.9 Required R.18.1 Required R.18.1 Required R.18.1 Required R.18.2 Required R.18.3 Required R.18.4 Required R.18.4 Required R.18.5 Required R.18.6 Required R.18.6 Required R.18.6 Required R.18.7 Required R.18.7 Required Required R.18.8 Required R.18.8 Required R.18.9 Required R.18.9 Required R.18.9 Required R.18.1 Required R.18.1 Required R.18.1 Required R.18.2 Required R.18.3 Required R.18.4 Required R.18.5 Required R.18.6 Required R.18.6 Required R.18.6 Required R.18.7 Required R.18.7 Required R.18.8 Required Required R.18.8 Required R.18.9 Required R.18.9 Required R.18.9 Required R.18.9 Required R.18.0 Required R.18.1 Required R.18.1 Required Required R.18.2 Required Required R.18.3 Required Required Required Required Required Required Required R.18.6 Required Required Required Required R.18.7 Required R.18.8 Required Requ		·	·		reassigned.
R.18.1 Required A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand R.18.2 Required Required R.18.3 Required R.18.4 Required A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand Subtraction between pointers shall only be applied to pointers that address elements of the same array The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object R.18.4 Advisory R.18.5 Advisory R.18.6 Required A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand shall address an element of the same array too big at call. 69 X Parameter indexing array too big at call. 68 X Parameter indexing array too small at call. 438 S Pointer arithmetic is not on array. Pointer subtraction not addressing one array. Pointer subtraction not addressing one array. Pointer subtraction not addressing one array. Pointer arithmetic is not on					
R.18.1 Required Poperand shall address an element of the same array as that pointer operand shall address an element of the same array as that pointer operand R.18.2 Required Subtraction between pointers shall only be applied to pointers that address elements of the same array R.18.3 Required be applied to objects of pointer type except where they point into the same object R.18.4 Advisory R.18.5 Advisory R.18.6 Required Required R.18.6 Required R.18.7 Required R.18.7 Required R.18.7 Required R.18.7 Required R.18.7 Required R.18.8 Required R.18.8 Required R.18.8 Required R.18.9 R				436 S	Declaration does not specify an array.
R.18.1 Required operand shall address an element of the same array as that pointer operand operand shall address an element of the same array as that pointer operand		Required	operand shall address an element of the same	567 S	Pointer arithmetic is not on array.
Array as that pointer operand array as that pointer operand 64 X Array bound exceeded at call. 68 X Parameter indexing array too big at call. 69 X Global array bound exceeded at use. 72 X Parameter indexing array too small at call. 73 X Parameter indexing array too small at call. 74 Pointer subtraction not addressing one array. 75 Pointer subtraction not addressing one array. 76 Pointer subtraction not addressing one array. 77 S Pointer subtraction not addressing one array. 87 S Use of pointer tobject pointers. 88 S Pointer arithmetic. 89 X Parameter indexing array too big at call. 89 X Parameter indexing array too big at call. 80 S Pointer subtraction not addressing one array. 80 S Pointer arithmetic. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer returned in function result. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer indirection exceeds 2 levels. 80 S Pointer arithmetic is not on array. 80 S Pointer indirection excee	D 10 1			692 S	Array index is negative.
R.18.2 Required Subtraction between pointers shall only be applied to pointers that address elements of the same array R.18.3 Required The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object R.18.4 Advisory The +, -, += and -= operators should not be applied to an expression of pointer type R.18.5 Advisory Required R	13.10.1			64 X	Array bound exceeded at call.
R.18.2 Required Subtraction between pointers shall only be applied to pointers that address elements of the same array The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object R.18.4 Advisory The +, -, += and -= operators should not be applied to an expression of pointer type R.18.5 Advisory Declarations should contain no more than two levels of pointer nesting R.18.6 Required The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist To be applied to an expression of pointer type R.18.6 Required Flexible array members shall not be declared The same object indexing array too small at call. 438 S Pointer subtraction not addressing one array. Pointer subtraction not addressing array on array. Pointer subtraction not addressing array.				68 X	Parameter indexing array too big at call.
R.18.2 Required Subtraction between pointers shall only be applied to pointers that address elements of the same array R.18.3 Required Declarations operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object R.18.4 Advisory The +, -, += and -= operators should not be applied to an expression of pointer type R.18.5 Advisory Declarations should contain no more than two levels of pointer nesting R.18.6 Required Required Pequired Subtraction between pointers shall not be declared A38 S Pointer subtraction not addressing one array. 437 S <> =>= used on different object pointers. 87 S Use of pointer arithmetic. 567 S Pointer arithmetic is not on array. 80 S Pointer indirection exceeds 2 levels. 42 D Local pointer returned in function result. 77 D Local structure returned in function result. 71 S Pointer assignment to wider scope. 565 S Assignment to wider scope.				69 X	Global array bound exceeded at use.
R.18.2 Required to pointers that address elements of the same array The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object R.18.4 Advisory R.18.5 Advisory Required Required Required The +, -, += and -= operators should not be applied to an expression of pointer type Declarations should contain no more than two levels of pointer nesting R.18.6 Required Required The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist R.18.7 Required The relational operators >, >=, < and <= shall not be except where they pointer type except where they point into the same object A37 S 437 S 438 S Pointer subtraction not addressing one array. 437 S 437 S 437 S 437 S Pointer arithmetic. 567 S Pointer indirection exceeds 2 levels. Pointer indirection exceeds 2 levels. 77 D Local structure returned in function result. 71 S Pointer assignment to wider scope. 565 S Assignment to wider scope. Assignment to wider scope.				72 X	Parameter indexing array too small at call.
R.18.3 Required be applied to objects of pointer type except where they point into the same object R.18.4 Advisory The +, -, += and -= operators should not be applied to an expression of pointer type R.18.5 Advisory Declarations should contain no more than two levels of pointer nesting The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist R.18.7 Required Declarations should contain no more than two levels of pointer nesting The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist Advisory R.18.6 Required R.18.7 Required Declarations should contain no more than two levels of pointer arithmetic. 87 S Use of pointer arithmetic. 88 S Pointer indirection exceeds 2 levels. 42 D Local pointer returned in function result. 77 D Local structure returned in function result. 71 S Pointer assignment to wider scope. 565 S Assignment to wider scope. 565 S Assignment to wider scope.	R.18.2	Required	to pointers that address elements of the same array	438 S	Pointer subtraction not addressing one
R.18.5 Advisory Declarations should contain no more than two levels of pointer nesting R.18.6 Required Required Required R.18.7 Required Required R.18.7 Required Placeholder R.18.6 Required Required R.18.7 Required Placeholder R.18.6 Required Placeholder R.18.6 Required Placeholder R.18.7 Required R.18.7 Required Placeholder R.18.7 Req	R.18.3	Required	be applied to objects of pointer type except where they point into the same object		, ·
R.18.5 Advisory Declarations should contain no more than two levels of pointer nesting The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist R.18.7 Required Advisory Declarations should contain no more than two levels of pointer nesting The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall be copied to another object that persists after the first object has ceased to exist The address of an object with automatic storage shall be copied to another object that persists after the first object has ceased to exist at the first object has cea	R 18 4	Advisory			
R.18.5 Advisory levels of pointer nesting The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist Required R.18.7 Required Idea of pointer nesting The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist R.18.7 Required Flexible array members shall not be declared R.18.8 Array with no bounds in struct.	11.10.4	Advisory		567 S	Pointer arithmetic is not on array.
R.18.6 Required Shall not be copied to another object that persists after the first object has ceased to exist R.18.7 Required Flexible array members shall not be declared 77 D Local structure returned in function result. 71 S Pointer assignment to wider scope. 565 S Assignment to wider scope. 481 S Array with no bounds in struct.	R.18.5	Advisory		80 S	
R.18.6 Required shall not be copied to another object that persists after the first object has ceased to exist R.18.7 Required Shall not be copied to another object that persists after the first object has ceased to exist 77 D Local structure returned in function result. 77 D Local structure returned in function result. 77 D Local structure returned in function result. 78 Pointer assignment to wider scope. 565 S Assignment to wider scope. 565 S Array with no bounds in struct.	R.18.6	Required	The address of an object with automatic storage		Local pointer returned in function result.
after the first object has ceased to exist R.18.7 Required Flexible array members shall not be declared 71.5 Pointer assignment to wider scope. 565.5 Assignment to wider scope. 481.5 Array with no bounds in struct.			shall not be copied to another object that persists	77 D	Local structure returned in function result.
R.18.7 Required Flexible array members shall not be declared 481 S Array with no bounds in struct.				71 S	Pointer assignment to wider scope.
			arter the first object has ceased to exist	565 S	Assignment to wider scope.
	R.18.7	Required	Flexible array members shall not be declared		
	R.18.8			621 S	

				String function params access same
R.19.1		An object shall not be assigned or copied to an	480 S	variable.
	Mandatory	overlapping object	545 S	Assignment of overlapping storage.
			647 S	Overlapping data items in memcpy.
R.19.2	Advisory	The union keyword should not be used	74 S	Union declared.
11.10.2	Advisory		75 S	Executable code before an included file.
R.20.1	Advisory	#include directives should only be preceded by preprocessor directives or comments	338 S	#include preceded by non preproc directives.
R.20.2	Required	The ', " or \ characters and the /* or // character sequences shall not occur in a header file name	100 S	#include filename is non conformant.
R.20.3	Required	The #include directive shall be followed by either a <filename> or "filename" sequence</filename>	427 S	Filename in #include not in < > or " ".
		A macro shall not be defined with the same name	86 S	Attempt to define reserved word.
R.20.4	Required	as a keyword	580 S	Macro redefinition without using #undef.
		ac a normana	626 S	#define of keyword.
R.20.5	Advisory	#undef should not be used	68 S	#undef used.
	7.00.00.7		426 S	#undef used in a block.
R.20.6	Required	Tokens that look like a preprocessing directive shall not occur within a macro argument	341 S	Preprocessor construct as macro parameter.
R.20.7	Required	Required Expressions resulting from the expansion of macro	78 S	Macro parameter not in brackets.
11.20.7		parameters shall be enclosed in parentheses	361 S	Expression needs brackets.
R.20.8	Required	The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1	616 S	Preprocessor result not 0 or 1.
R.20.9	Required	All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation	337 S	Undefined macro variable in #if.
R.20.10	Advisory	The # and ## preprocessor operators should not be used	125 S	Use of ## or # in a macro.
R.20.11	Required	A macro parameter immediately following a # operator shall not immediately be followed by a ## operator	637 S	# operand followed by ##.
R.20.12	Required	A macro parameter used as an operand to the # or ## operators, which is itself subject to further macro replacement, shall only be used as an operand to these operators	125 S	Use of ## or # in a macro.
R.20.13	Required	A line whose first token is # shall be a valid	147 S	Spurious characters after preprocessor directive.
	,	preprocessing directive	342 S	Extra chars after preprocessor directive.

				I
R.20.14	Required	All #else, #elif and #endif preprocessor directives	126 S	A #if has no #endif in the same file.
11.20111	rtoquirou	shall reside in the same file as the #if, #ifdef or	343 S	#else has no #if, etc in the same file.
		#define and #undef shall not be used on a	86 S	Attempt to define reserved word.
R.21.1	Required	reserved identifier or reserved macro name	156 S	Use of 'defined' keyword in macro body.
		reserved identifier of reserved macro name	219 S	User name starts with underscore.
D 04 0		A reserved identifier or reserved macro name shall	218 S	Name is used in standard libraries.
R.21.2	Required	not be declared	219 S	User name starts with underscore.
D 04 0		The memory allocation and deallocation functions		
R.21.3	Required	of <stdlib.h> shall not be used</stdlib.h>	44 S	Use of banned function, type or variable.
		The standard header file <setjmp.h> shall not be</setjmp.h>		
R.21.4	Required	used	43 S	Use of setjmp/longjmp.
		The standard header file <signal.h> shall not be</signal.h>		
R.21.5	Required	used	130 S	Included file is not permitted.
		The Standard Library input/output routines shall	44 S	Use of banned function, type or variable.
R.21.6	Required	not be used.	130 S	Included file is not permitted.
		The Standard Library functions atof, atoi, atol and	130 3	Included file is not permitted.
R.21.7	Required	atoll of <stdlib.h> shall not be used</stdlib.h>	44 S	Use of banned function, type or variable.
		The Standard Library functions abort, exit and	44.0	Llac of hanned function, tune or verichle
R.21.8	Required	system of <stdlib.h> shall not be used</stdlib.h>	44 S	Use of banned function, type or variable.
	·	•	122 S	Use of abort, exit, etc.
R.21.9	Required	The Standard Library functions bsearch and qsort	44 S	Use of banned function, type or variable.
	'	of <stdlib.h> shall not be used</stdlib.h>		
R.21.10	Required	The Standard Library time and date routines shall	44 S	Use of banned function, type or variable.
		not be used	130 S	Included file is not permitted.
R.21.11	Required	The standard header file <tgmath.h> shall not be</tgmath.h>	130 S	Included file is not permitted.
14.21.11	rtoquilou	used	100 0	moraded me to not permitted.
R.21.12	Advisory	The exception handling features of <fenv.h></fenv.h>	44 S	Use of banned function, type or variable.
11.21.12	Advisory	should not be used	77 0	Ose of barried function, type of variable.
		Any value passed to a function in <ctype.h> shall</ctype.h>		Invalid value may be passed to function in
R.21.13	Mandatory	be representable as an unsigned char or be the	663 S	· ·
	,	value EOF		<ctype.h>.</ctype.h>
D 04 44	5	The Standard Library function memcmp shall not	004.0	memcmp used to compare null terminated
R.21.14	Required	be used to compare null terminated strings	661 S	strings.
		The pointer arguments to the Standard Library		Ĭ
D • · · -		functions memcpy, memmove and memcmp shall	A== -	Standard library copy/compare objects have
R.21.15	Required	be pointers to qualified or unqualified versions of	655 S	different types.
		compatible types		J. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
		companio types		

R.21.16	Required	The pointer arguments to the Standard Library function memcmp shall point to either a pointer type, an essentially signed type, an essentially unsigned type, an essentially Boolean type or an essentially enum type	618 S	Use of memcmp between structures.
		Use of the string handling functions from	600 S	Insufficient space for operation. Argument of strlen is unterminated. Copy source parameter not checked before
R.21.17	Mandatory	<string.h> shall not result in accesses beyond the bounds of the objects referenced by their pointer parameters</string.h>	140 D 66 X 70 X	use. Insufficient array space at call. Array has insufficient space.
			71 X	Insufficient space for copy. Insufficient space for operation.
		The size 4 annument page of the size formation :	66 X	Insufficient array space at call.
R.21.18	Mandatory	The size_t argument passed to any function in	70 X	Array has insufficient space.
		<string.h> shall have an appropriate value</string.h>	71 X	Insufficient space for copy.
			79 X	Size mismatch in memcpy/memset.
R.21.19	Mandatory	The pointers returned by the Standard Library functions localeconv, getenv, setlocale or, strerror shall only be used as if they have pointer to const-qualified type	107 D	Attempt to change system call capture string.
R.21.20	Mandatory	The value returned by a call of one of the Standard Library functions asctime, ctime, gmtime, localtime, localeconv, getenv, setlocale or strerror shall not be used following a subsequent call to the same function	133 D	Pointer from system function used after subsequent call.
		All resources obtained dynamically by means of	49 D	File pointer not closed on exit.
R.22.1	Required	Standard Library functions shall be explicitly released	50 D	Memory not freed after last reference.
R.22.1				Attempt to open file pointer more than once.
R.22.2	Mandatory		51 D	Attempt to read from freed memory.
			125 D	free called on variable with no allocated space.
		A block of memory shall only be freed if it was	407 S	free used on string.
		allocated by means of a Standard Library function		Freed parameter is not heap item.
			484 S	Attempt to use already freed object.
			644 S	realloc ptr does not originate from allocation function.

R.22.3	Required	The same file shall not be open for read and write access at the same time on different streams	103 D	File opened both read and write.
R.22.4	Mandatory	There shall be no attempt to write to a stream which has been opened as read-only	98 D	Attempt to write to file opened read only.
R.22.5	Mandatory	A pointer to a FILE object shall not be dereferenced	591 S	Inappropriate use of file pointer.
R.22.6	Mandatani	The value of a pointer to a FILE shall not be used	48 D	Attempt to write to unopened file.
K.22.0	Mandatory	after the associated stream has been closed	113 D	File closed more than once.
R.22.7	Required	The macro EOF shall only be compared with the unmodified return value from any Standard Library function capable of returning EOF	662 S	EOF compared with char.
R.22.8	Required	The value of errno shall be set to zero prior to a	111 D	errno checked without having been set for errno setting fn.
11.22.0	Required	call to an errno-setting-function	121 D	errno neither set nor checked for errno setting function.
			121 D	errno neither set nor checked for errno setting function.
R.22.9	Required	The value of errno shall be tested against zero after calling an errno-setting-function	122 D errno not checked after being set f setting fn.	errno not checked after being set for errno setting fn.
			134 D	errno not checked before subsequent function call.
R.22.10	Required	The value of errno shall only be tested when the last function to be called was an errno-setting-function	132 D	errno checked after call to non-errno setting function.

General Compliance Notes

Enhanced Enforcement: LDRA checks additional cases to those specified by the mapped rule for enhanced safety and security.

Fully Implemented: LDRA checks all statically checkable aspects of the mapped rule.

Partially Implemented: LDRA checks certain aspects of the rule.

The assessment of whether a rule is fully or partially implemented is based on whether the mapped LDRA standards cover all statically checkable aspects of the rule with a high level of coverage or only cover certain statically checkable aspects of the rule. If a rule is undecidable then this assessment is based on what it is deemed reasonable for a static analysis tool to check.