Exercise 7 - MISRA standard

New Attempt

Due No Due Date **Points** 1 **Submitting** a file upload

To set up for this exercise:

- In CS50 IDE make new folder *lab07*
- To check for code conformance to MISRA standard we will use compiler for Texas Instruments C6000 ("TI compiler" in the further text).
  - Download the compiler **tool**, extract it and copy in the IDE.
  - Position in the appropriate path, where you copied the files, in the terminal. Go to *ti_tools* folder and run *source set_env.sh* (Note: you need to do this step whenever you open a new terminal)
  - Script from the previous step give appropriate permissions to tools and sets up system path so that the tool can be used from any path.
  - Here you can find detailed manual for the TI compiler **here**. Command line switched are given in chapters 2.2 and 2.3.1, whereas description of MISRA check is given in 6.3.
- Calling the TI compiler is just a set in code development. To actually build the executable you still need to compile it with GCC. Output of TI compiler (object file) is irrelevant, only the output that TI compiler gives in console. It represents the report about code conformance with MISRA.
- As the result of this whole exercise upload *program.c,* after you satisfied all the rules that you wanted or was able to satisfy, as well as *list_of_broken_rules.txt* that should contain the list of rules that remain unsatisfied.

With the TI compiler check *program.c* from Excercise 1, Task 1 for MISRA conformance. (Note: the TI compiler is checking against MISRA 2004, whereas the latest standard is MISRA 2012. Also, the TI compiler's support for MISRA checking is experimental and does have some issues. Still, it is good enough for ilustrating MISRA in this exercise.)

You can call the tool with : *cl6x --include_path=$TI_INCLUDE --compile_only --check_misra=all program.c*

If you set everything properly, you should see a lot of text output, with bunch of reports about MISRA nonconformance, regarding different rules. Focus just on reports related to *program.c*, ignore the ones that are related to system and standard library headers (most of the issues are actually related to those, so do not be taken back by the amount of dumped text). Think about every broken rule and how to change the code so you can satisfy that rule.

Try to change the code so to satisfy as most MISRA rules as you can (i.e. to have as few rules that you do not satisfy). To fully understand the rules these documents can be helpful to you:

- "MISRA-C:2004 - Guidelines for the use of the C language in critical systems" - MISRA 2004 standard itself
- "MISRA C 2004 Permits" - List of rules that can be broken and the conditions in which it is allowed
- "MISRA C 2012 Addendum 1 - Rule Mapping" - Table of rule mapping from MISRA 2004 to MISRA 2012 (the latest), with explanations why some rules where changed (strengthen, relaxed, removed, or why some new ones where introduced)
- In case that the above documents are not available to you, at least see MISRA 2004 rules list with very short descriptions in "IAR Embedded Workbench MISRA C:2004 Reference Guide" **here (https://wwwfiles.iar.com/arm/webic/doc/EW_MisraC2004Reference.ENU.pdf)** .

If you can, in MISRA 2004 standard read the description of every broken rule, and than try to find it in the other two MISRA documents mentioned above.

Pay attention to these rules:

- 2.2 - In the newest MISRA this rule is changed from required to advisory. If you can, analyze reasons for introduction of that rule, and then reasons for removing it form the list of required rules. Decide whether you want to follow that rule.
- 5.7 - Removed from MISRA 2012. Think about this rule and whether you want to conform to it or not.
- 8.1 - Relaxed in MISRA 2012. If you can, analyze what was changed in this rule between 2004 and 2012.
- 17.4 - Relaxed in MISRA 2012 changed from required to advisory. If you can, analyze the differences and think about them. Remember different was that you can declare that function can accept array (or pointer) as a parameter - that holds key to satisfying this rule.
- 20.9 - Is it possible to satisfy this rule in this code?

For some rules, a legitimate conclusion can be that we do not want or can not satisfy them.

For parts of the code you can turn off MISRA check using *#pragma CHECK_MISRA("none").* To turn on checking again, use *#pragma CHECK_MISRA("all").* You can use this to turn of check for all the standard headers (all #includes of standard headers), reduce the amount of text you get as output.

Checking for a specific rule can be turned of with *#pragma CHECK_MISRA("-X")* where X is the rule number. To turn the check back on again: *#pragma CHECK_MISRA("X").*

More on this you can see in the TI compiler manual (chapters 6.3, 6.9.1 and 6.9.25).

Additional task

Play a bit and analyze for MISRA conformance some other code that you have written during this course.