C Programming Exercise

These are some general tasks or exercises that exploit different principles found in C programming language when used on Embedded Systems (or in general). Exercises are categorised. Here are a couple of pointers (pun intended):

- As solution provide:
    - Only .c and (if needed) .h files in a folder named: <category>_<exercise number>
    - For text answers in .txt file in a folder named <category>_<exercise number>
- Solutions should not include compiler specific macros or pragmas (except if really needed)
- Solution should be compilable with the latest GCC compiler.
- Be aware to cover edge cases, make your exercise solution more robust.
- Speed of solution making is not as important as correctness. So rather take some more time to verify that your code does what exercise demands of it.

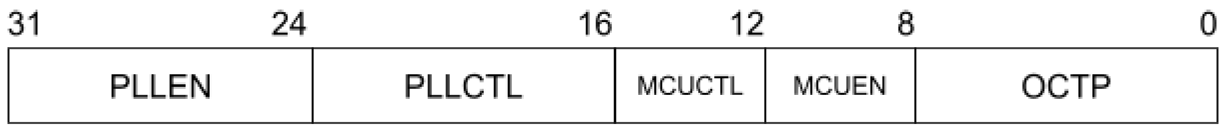
Structures

1. You and your friends Don and Glenn have an argument about which of your three cars is the best. To settle this, you decide to use your programming skills, so: create a struct(ure) named `carInstance`, which has following structure members:
   - `maxSpeed`: which is used to store car's maximum speed
   - `maxRPM`: which is used to store car's maximum RPM
   - `torque`: which is used to store car's torque
   - `horsepower`: which is used to store car's horsepower
   - `numberOfGears`: : which is used to store car's number of gears

   Then in code create three instances of carInstance, for you and each of your friends. Create then function(s) that would print to console, which of the three cars is best for a specific category (struct member).

2. Create the same solution as in 1) that settles disputes between different numbers of friends, not just three (by only adding new structure instances in code).

3. We have an embedded system with 32-bit register SYS_CTRL, that has following structure:

   So, to change:
   - OCTP bits, we need to change bits 0-7 in SYS_CTRL register
   - MCUCTL bits, we need to change bits 8-11 in SYS_CTRL register
   - …

   How would you define a structure in C that would change exactly and only MCUCTL bits when one would write following code:

   ```
   SYS_CTRL.MCUCTL = 0x1;
   ```

Project with multiple source files

1. Create two .c files:
   - One that contains function `printParameter` (with one integer parameter), that outputs passed integer parameter to console:

     ```
     I printParameter, was just called with parameter <parameter
     value>
     ```

   - One that contains the main function with calls to the `printParameter` function from the other file.