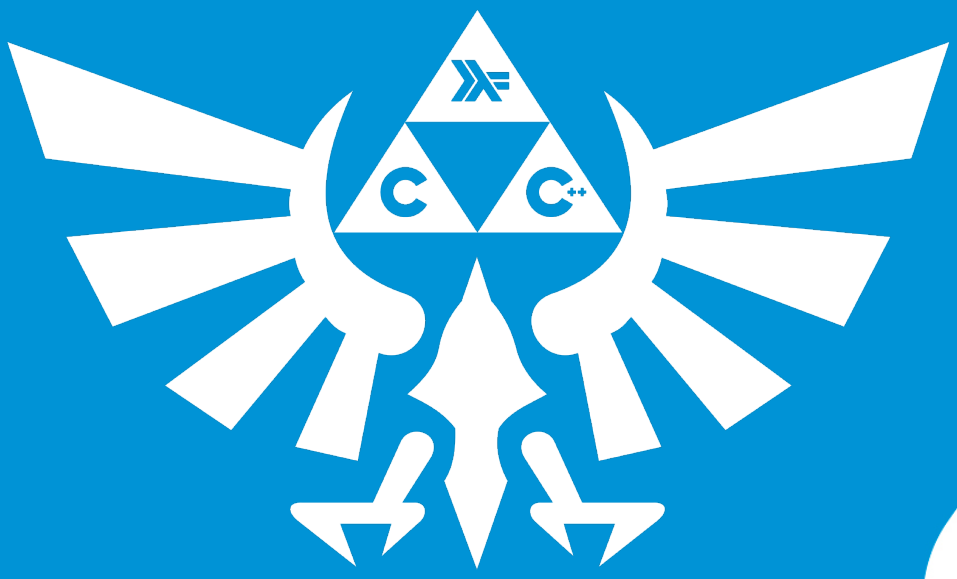
{EPITECH}

DAY 04 AM

C, LIFE, THE UNIVERSE AND EVERYTHING ELSE



DAY 04 AM

All your exercises will be compiled with the `-std=gnu17 -Wall -Wextra` **flags**, unless specified otherwise.

Every function implemented in a header or any unprotected header leads to 0 for the exercise.

To avoid compilation problems during automated tests, please include all necessary files within your headers.

For each exercise, the files must be turned-in in a separate directory called **exXX** where XX is the exercise number (for instance `ex01`), unless specified otherwise.

Unit Tests

It is highly recommended to test your functions as you implement them. It is common practice to create and use what are called **unit tests**.

From now on, we expect you to write unit tests for your functions (when possible). To do so, please follow the instructions in the **"How to write Unit Tests"** document on the intranet, available here.

Create a directory named `tests`. For each of the functions you turn in, create a file in that directory named `tests_FUNCTION_NAME.c` containing all the tests needed to cover all of the exercise's possible cases (regular or irregular).

Here is a sample set of unit tests for the **my_strlen** function :

```
#include <criterion/criterion.h>

Test(my_strlen, positive_return_value)
{
        cr_assert_eq(my_strlen("toto"), 4);
}

Test(my_strlen, empty_string)
{
        cr_assert_eq(my_strlen(""), 0);
}
```

Exercise 0 - Z is for Zorglub

Turn in : `Makefile`, `z.c` in `ex00/`
Compilation : using your `Makefile`
Executable name : `z`

Our Lord and Genius Zorglub, master of Zorgland, wants you to design the software for his new brain-washing machine to zorgmanize his enemies into zorgmen. The machine works by sending a specific character into the mind of its victim. The character must be chosen carefully using an algorithm devised by our Lord and Genius Zorglub.

The algorithm uses the prisoner ID to determine which character should be used. An ID is a hexadecimal number that fits in a `uint64_t`, given as parameter to your program. You must then display the character to be used, following the rules defined by the Grand Z :

- ✓ If the binary notation of the ID is a palindrome, it means that the victim is just a dummy used for test purpose. You must display the default brain-washing byte `0172` (octal) and validation byte `012` (octal).
- ✓ If the ID is a multiple of 13 and 29 and 89 or 41 and 71 or 67 or 7 and 43 and 47 and 53, you must display the string `"z\n"` and immediately stop your program.
- ✓ If the ID is `0x12345678`, `0x87654321`, `0x01111010` or `0x01011010`, you must display the following bits : `0111101000001010`.
- ✓ If one of the byte of the ID is worth `0x42`, it means that the victim is tough and the strongest character must be used : `'z'`. The byte `0x0A` is then displayed.
- ✓ If the last byte of the ID is null, the victim is just a minion of our enemies. As we don't care about them, we will brain-wash them using the last letter of the alphabet in lowercase. The output will then be flushed with a line feed.
- ✓ If the ID is a multiple of the sum of its 8 bytes, you must display the sixth character of the ASCII table starting from the end, followed by a `'\n'`.
- ✓ If the 8 bytes of the ID are identical, it means that the prisoner is one of our special guest. You will use the character `'Z'` in lowercase, and your program will end with a new line. This case override all the other cases.
- ✓ If the parameter contains non-hexadecimal characters, this is an error. The program must handle this case by displaying the error character `'z'` followed by a line feed.
- ✓ If the parameter does not fit in a `uint64_t`, this is an error. In this very specific case, the very specific character `'z'` must be used to brain-wash the victim, followed by the tenth character of the ASCII table.
- ✓ If there is no argument, the ID to be used is the return value of the function `time(NULL)`.
- ✓ If there is too many argument, your program must use the first valid argument starting from the end. If no argument is valid, we consider there is no argument and use the preceding rule.