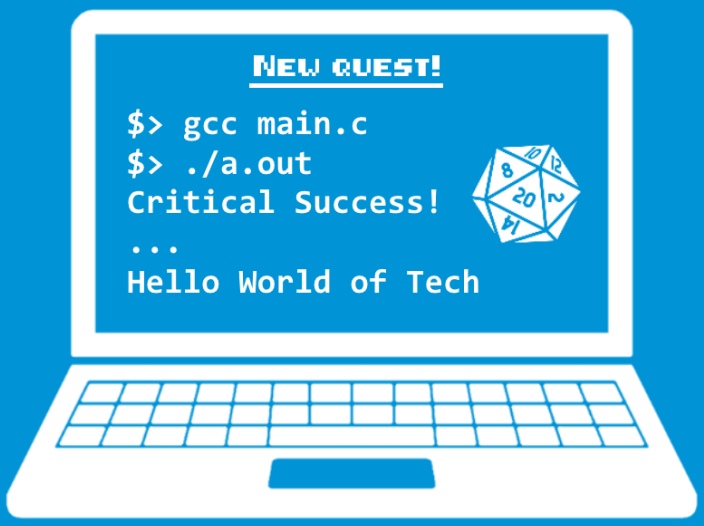
{EPITECH}

# DAY 04

## POINTERS



# DAY 04

## Preliminaries

**Language:** C

The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

- ✓ Don't push your `main` function into your delivery directory, we will be adding our own. Your files will be compiled adding our `main.c` and our `my_putchar.c` files.
- ✓ You are only allowed to use the `my_putchar` function to complete the following tasks, but **don't push it** into your delivery directory, and don't copy it in *any* of your delivered files.
- ✓ If one of your files prevents you from compiling with `*.c`, the Autograder will not be able to correct your work and you will receive a 0.

Clone your repository at the beginning of the day and submit your work on a regular basis!
The delivery directory is specified within the instructions for each task.
In order to keep your repository clean, pay attention to `gitignore`.

## Task 01 - my_swap

**Delivery:** my_swap.c

Write a function that swaps the content of two integers, whose addresses are given as a parameter. It must be prototyped as follows:

```
void my_swap(int *a, int *b);
```

## Task 02 - my_putstr

**Delivery:** my_putstr.c

Write a function that displays, one-by-one, the characters of a string.

The address of the string's first character will be found in the pointer passed as a parameter to the function, which must be prototyped as follows:

```
int my_putstr(char const *str);
```

## Task 03 - my_strlen

**Delivery:** my_strlen.c

Write a function that counts and returns the number of characters found in the string passed as parameter. It must be prototyped as follows:

```
int my_strlen(char const *str);
```

## Task 04 - my_evil_str

**Delivery:** my_evil_str.c

The goal of this task is to swap each of the string's characters, two by two.
In other words, you will swap the first letter with the last one, the second with the second-to-last and so on.
The function should return a pointer to the first character of the reversed string:

```
char *my_evil_str(char *str);
```

For instance

```
a => a
ab => ba
abc => cba
abcd => dcba
abcde => edcba
abcdef => fedcba
```

When testing your function you may encounter "Segmentation fault" errors. Either you're messing with the pointers in your function or the string given in parameter is read-only!

Easy way to have read/write string for testing purpose: `man 3 strdup`.

## Task 05 - my_getnbr

**Delivery:** my_getnbr.c

Write a function that returns a number, sent to the function as a string. It must be prototyped as follows:

```
int my_getnbr(char const *str);
```

Here are some tricky examples to be handled:

```
"+---+--++---+---+---+-42" => -42
"42a43" => 42
"11000000000000000000000042" => 0 (the number doesn't fit in an integer)
"-10000000000000000000000042" => 0 (for the same reason}
```

## Task 06 - my_sort_int_array

**Delivery:** my_sort_int_array.c

Write a function that sorts an integer array in ascending order, given a pointer to the first element of the array and its size.

```
void my_sort_int_array(int *array, int size);
```

v 3.1