{EPITECH}

DAY 07

< LIBMY, ARGUMENTS />

DAY 07

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`.
- ✓ 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.

All .c files from your delivery folder will be collected and compiled with your `libmy`, which must be found in `lib/my/`. For those of you using .h files, they must be located in `include/` (like the `my.h` file).

Some tests will automatically compile your functions the following way:

```
Terminal
$>  cd taskXX
$> clang *.c -c -I../include/
$> clang *.o autograder/main_taskXX.o -L../lib/my/ -o taskXX -lmy
```

Your library will be built using the `lib/my/build.sh` script you will create in the first task.

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`.

Allowed system function(s): `write`

We still encourage you to write unit tests for all your functions!
Check out Day06 if you need an example, and re-read the guide.

Task 01 - libmy.a

Delivery: lib/my/build.sh

Create a shell script that, when executed, build your own library in `lib/my/` and name it `libmy.a`.
The library **MUST** contain **ALL** of the following functions:

```
1 void my_putchar(char c);
2 int my_isneg(int nb);
3 int my_put_nbr(int nb);
4 void my_swap(int *a, int *b);
5 int my_putstr(char const *str);
6 int my_strlen(char const *str);
7 int my_getnbr(char const *str);
8 void my_sort_int_array(int *tab, int size);
9 int my_compute_power_rec(int nb, int power);
10 int my_compute_square_root(int nb);
11 int my_is_prime(int nb);
12 int my_find_prime_sup(int nb);
13 char *my_strcpy(char *dest, char const *src);
14 char *my_strncpy(char *dest, char const *src, int n);
15 char *my_revstr(char *str);
16 char *my_strstr(char *str, char const *to_find);
17 int my_strcmp(char const *s1, char const *s2);
18 int my_strncmp(char const *s1, char const *s2, int n);
19 char *my_strupcase(char *str);
20 char *my_strlowcase(char *str);
21 char *my_strcapitalize(char *str);
22 int my_str_isalpha(char const *str);
23 int my_str_isnum(char const *str);
24 int my_str_islower(char const *str);
25 int my_str_isupper(char const *str);
26 int my_str_isprintable(char const *str);
27 int my_showstr(char const *str);
28 int my_showmem(char const *str, int size);
29 char *my_strcat(char *dest, char const *src);
30 char *my_strncat(char *dest, char const *src, int nb);
```

Beware to build your `libmy.a` library in the correct folder because it will be used to compile all of your programs.
All the source code used to build the library must be present in your `lib/my/` directory on your repository.
Do NOT add the built `libmy.a` in your repository!

The functions from the following two tasks must be included in your library.
From tomorrow onwards, none of the functions present in your library must be present in your sources.

Task 02 - my_strcat

Delivery: my_strcat.c

Write a function that concatenates two strings. It must be prototyped the following way:

```
char *my_strcat(char *dest, char const *src);
```

```
man strcat
```

Task 03 - my_strncat

Delivery: my_strncat.c

Write a function that concatenates `n` characters of the `src` string to the end of the `dest` string.
It must be prototyped the following way:

```
char *my_strncat(char *dest, char const *src, int nb);
```

Task 04 - my_print_params

Delivery: task04/*.c

Write a program that displays its arguments (received on the command line). Since it is a **PROGRAM**, you need to put the `main` function in your delivered files.
You are to display all arguments (including `argv[0]`), on different lines.

Your main function must return `0`.

Terminal

```
$>  ./a.out test "This is a test " retest | cat -e
./a.out$
test$
This is a test $
retest$
```

Task 05 - my_rev_params

Delivery: task05/*.c

Write a program that displays all the arguments received on the command line in reverse order.
You are to display all arguments (including `argv[0]`), on different lines.

Your main function must return `0`.

Terminal

```
$>  ./a.out test "This is a test " retest | cat -e
retest$
This is a test $
test$
./a.out$
```

Task 06 - my_sort_params

Delivery: task06/*.c

Write a program that displays all its arguments, in `ascii` order.
You are to display all arguments (including `argv[0]`), on different lines.

Your main function must return `0`.

Terminal

```
$>  ./a.out test "This is a test " retest | cat -e
./a.out$
This is a test $
retest$
test$
```

v 3.1