B1- Unix and C Lab Seminar

B-CPE-100

Day 03

First C Programming

v1.7

EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

Day 03

First C Programming

repository name:	CPool_Day03_$ACADEMICYEAR
repository rights:	ramassage-tek
language:	C
group size:	1

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- 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.

Arrays are **forbidden** for every task.

Create 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 00

Coding Style

At Epitech, every C code must comply with our Coding Style: epitech_coding_style.pdf

Read it carefully! The code quality is an important factor and will be evaluated!

To help you with the basics of following this coding style, we have prepared simple configuration files for emacs. You can download the archive here, read the README file within and follow its instructions.

Task 01

my_print_alpha

Write a function that, beginning with **a**, displays the lowercase alphabet in ascending order, on a single line. It must be prototyped as follows:

```
int my_print_alpha(void);
```

Delivery:: CPool_Day03_$ACADEMICYEAR/my_print_alpha.c

Task 02

my_print_revalpha

Write a function that, beginning with **z**, displays the lowercase alphabet in descending order, on a single line. It must be prototyped as follows:

```
int my_print_revalpha(void);
```

Delivery: CPool_Day03_$ACADEMICYEAR/my_print_revalpha.c

Task 03

my_print_digits

Write a function that displays all the digits, on a single line, in ascending order. It must be prototyped as follows:

```
int my_print_digits(void);
```

Delivery: CPool_Day03_$ACADEMICYEAR/my_print_digits.c

Task 04

my_isneg

Write a function that displays either **N** if the integer passed as parameter is negative, **P**, if positive or null. It must be prototyped as follows:

```
int my_isneg(int n);
```

Delivery: CPool_Day03_$ACADEMICYEAR/my_isneg.c

Task 05

my_print_comb

Write a function that displays, in ascending order, all the numbers composed by three *different* digits numbers (012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 789). Given three digits (all different), only the smallest number composed by those digits must be displayed. It must be prototyped as follows:

```
int my_print_comb(void);
```

Delivery: CPool_Day03_$ACADEMICYEAR/my_print_comb.c

```
Terminal
~/B-CPE-100> ./a.out
012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 789
```

Neither **987** nor **999** is to be displayed (as an example).

Task 06

my_print_comb2

Write a function that displays, in ascending order, all the different combinations of two two-digit numbers (00 01, 00 02, 00 03, 00 04, 00 05,...,01 99, 02 03, ..., 98 99). It must be prototyped as follows:

```
int my_print_comb2(void);
```

Delivery: CPool_Day03_$ACADEMICYEAR/my_print_comb2.c

```
Terminal
~/B-CPE-100> ./a.out
00 01, 00 02, 00 03, 00 04, 00 05,...,01 99, 02 03, ..., 98 99
```

Task 07

my_put_nbr

Write a function that displays the number given as a parameter. It must be able to display all the possible values of an **int**, and must be prototyped as follows:

```
int my_put_nbr(int nb);
```

Delivery: CPool_Day03_$ACADEMICYEAR/my_put_nbr.c

For instance, my_put_nbr(42) displays 42, my_put_nbr(0) displays 0, my_put_nbr(-2147483647) displays -2147483647.

Task 08

Testing

It is highly recommended to test your functions as you develop them. It is common practice to create a function named `main` (and a designated file to host it) to check the functions separately.
Create a directory named `tests`.
Create a `main` function within a file named `tests-my_put_nbr.c`, to be stored in the `tests` directory named.
This function must contain all the necessary calls to `my_put_nbr` in order to cover all of the function's possible situations (both regular or irregular).

For instance, for the `my_isneg` function, you could have a file similar to the following:

```
int main()
{
    my_isneg(0);
    my_isneg(21);
    my_isneg(-21);
    return (0);
}
```

Testing is an important part of software development. There is no excellence without testing.

Task 09

my_print_combn

Write a function that displays, in ascending order, all the numbers composed by n *different* digits numbers (n being given as a parameter). Given n digits (all different), only the smallest number composed by thoses digits must be displayed. It must be prototyped as follows:

```
int my_print_combn(int n)
```

Delivery: CPool_Day03_$ACADEMICYEAR/my_print_combn.c

`my_print_combn(3)` gives the same result as `my_print_comb`.