# C - Pool - Tek1

## Subject Day 03

C Pool Managers
looneytunes@epitech.eu

# Contents

# Instructions

- The subject may change until one hour before turn-in.
- Respect the norm takes time, but is good for you. This way your code will respect the norm since the first written line.
- Ask yourself if it's relevant to let a `main()` function in your turn-in knowing we will add our own.
- We will compile your files with the command `cc *.c`, adding our `main.c` and our `my_putchar.c` :

```
$> cc *.c ~moulinette/main_ex_01.c ~moulinette/my_putchar.c -o ex01
$> ./ex01
[...]
```

- This is a turn-in directory, of course you will only keep in it your final work revision. No temporary file should stand there!
  You shall leave in your directory no other files than those explicitly specified by the exercises.
  If one of your files prevents the compilation with `*.c`, the robot will not be able to do the correction and you will have a `0`. That is why it's in your interest to remove any file that doesn't work.
- You are only allowed to use the `my_putchar` function to do the following exercises. This function will be provided, so:
  - You shall not have a `my_putchar.c` file in your turn-in directory.
  - The function `my_putchar` shall not be in any of your turned-in files.
- Don't forget to discuss about it in the pool section of the forum !
- For every exercise, arrays are forbidden !
- Turn-in directory:
  `Piscine_C_J03`

 *Hints* `Remember it is always better to create your repository at the beginning of the day and to turn-in your work on a regular basis`

 *Hints* `On the instructions of each exercises, this directory is specified for every turn-in path`

*Hints* In order to keep your working directory clean, we advise you to look carefully "gitignore" in order to avoid committing the .o files and the binaries

# Exercise 1 - my_aff_alpha

- Write a function that displays the alphabet in lowercase on a single line, in ascending order from the letter 'a'.
- The function shall be prototyped as follows:

```
1   int my_aff_alpha();
```

- Turn-in:
`Piscine_C_J03/my_aff_alpha.c`



# Exercise 2 - my_aff_revalpha

- Write a function that displays the alphabet in lowercase on a single line, in descending order from the letter 'z'.
- The function shall be prototyped as follows:

```
1   int my_aff_revalpha();
```

- Turn-in:
  `Piscine_C_J03/my_aff_revalpha.c`

# Exercise 3 - my_aff_chiffre

- Write a function that displays all the digits in a single line, in the ascending order.
- The function shall be prototyped as follows:

```
1   int my_aff_chiffre();
```

- Turn-in:
  `Piscine_C_J03/my_aff_chiffre.c`

# Exercise 4 - my_isneg

- Write a function that displays 'N' or 'P' depending on the sign of the integer passed as parameter. If n is negative, display 'N'. Otherwise, if n is positive or null then display 'P'
- The function shall be prototyped as follows:

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

- Turn-in:
`Piscine_C_J03/my_isneg.c`

# Exercise 5 - my_aff_comb

- Write a function that displays in the ascending order all the different combinations of three different digits in the ascending order.
- This gives something like that :
  “012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 789”
- 987 is not here because 789 is already there
- 999 is not here because that number’s digits are not all different from each other
- The function shall be prototyped as follows:

```
1   int my_aff_comb();
```

- Turn-in:
  `Piscine_C_J03/my_aff_comb.c`

# Exercise 6 - my_aff_comb2

- Write a function that displays all the different combinations of two numbers between 0 and 99, in the ascending order.
- This gives something like that :
  “00 01, 00 02, 00 03, 00 04, 00 05, ..., 01 99, 02 03, ..., 98 99”
- The function shall be prototyped as follows:

```
1   int my_aff_comb2();
```

- Turn-in:
  `Piscine_C_J03/my_aff_comb2.c`

# Exercise 7 - my_put_nbr

- Write a function that displays a number given as parameter. The function must be able to display all the possible values of an `int`.
- The function shall be prototyped as follows:

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

- Example:
  - my_put_nbr(42) displays “42”
  - my_put_nbr(0) displays “0”
  - my_put_nbr(-42) displays “-42”
  - my_put_nbr(2147483647) displays “2147483647”
  - my_put_nbr(-2147483648) displays “-2147483648”
- Turn-in:
  `Piscine_C_J03/my_put_nbr.c`

# Exercise 8 - Unit Tests

- It is highly recommended to test your functions when you are developing them.
- Usually, it is common to create a function named "`main`" (and a dedicated file to host it) to check the functions separately.
- Create a directory named "`tests`".
- Create a function "`int main()`" in a file named "`tests-my_put_nbr.c`", stored inside the directory "`tests`" previously created.
- According to you, this function must contains all the necessary call to "`my_put_nbr`" to cover all possible cases (special or regular) of the function.
- Thus, for the function named "`my_isneg`", we could have a file similar to:

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

- You have to compile your main with the file containing your function "`my_isneg`".

```
Exemple:
cc my_isneg.c tests/tests-my_isneg.c
```

- To conclude, we are inviting you to put all of your "`main`" into the "`tests`" directory of the day by respecting the following naming: "`tests-my_function.c`"
- The robot will indicate you the percentage of tests covered by your "`mains`".

`We are expecting you to do the same for all other days`

# Exercise 9 - my_aff_combn

- Write a function that displays all the combinations of n digit(s), in the ascending order.
- If n = 2, it shall display something like:
  “01, 02, 03, ..., 09, 12, ..., 79, 89”
- The function shall be prototyped as follows:

```
1   int my_aff_combn(int n);
```

- Turn-in:
  `Piscine_C_J03/my_aff_combn.c`