Premises

The objectiv of that Coding club is to give you some bases with the C language and a foretaste of the form, but also of the requiered programmation rigour, who will be asked for the next october.

And for a wonderful travel throught the day -1 of your pool in C you have to follow some rules !

Rules :

- Each exercise are maded in a way to be realised **only with the syscall funtction write()** (exepted if mentionned) of the library **unistd**. All extern fonction used in your exercise [*example printf(), puts()*] will cause an error in our moulinnette and will refused in any case your exercise (even if it is a correct one).
- You should organised your work in a tree structure of coherent folders to simplified the work of the moulinette. each name of each files and folders will be communicate at the begin of each exercise

Exemple of a tree:

```
.
├── my_name
│   └── my_name.c
└── digits
    └── digits.c
```

- If you have a question, ask at your right neighbor. Even if all efforts of your dear neighbore could solve your problem, ask by default your left neighbore. If you are at the end of a table so in the impossibility to have a right or left neighbore, it steal the one in front of you !
- If even all of that a question still not answered or a problem still unsolved, don't hesitate to call a cobra, they are here for that !
- All along of those exercises, we will ask you to realise **functions**. It's mean that our moulinette, will bring her own main() to test your work.

If you let, inadvertently your main() function, it's possible that it won't work and/or that could be a mistake too !

If you don't know what is a main() function ? Google it ;)

Moulinette

During the day, Cobras will lunch scripts of correction, who will let you know if an exersices is correct or not.

You would be able to check your result under the following form:

```
[Test Name]:  PASSED ou FAILED
```

This script, that we would call moulinette, is a programm who will correct your work.

It was carefully realised and can't be wrong... if an exercise fonction by you, but the moulinnette says the contrary, try to guess all possible cases following the test names to guess it. (We also do riddles in th Coding club, so fun) !

If really you think that the moulinnette is wrong. *(NB: Barely possible)*

Or even with **all your reflection** a test style an absolut mystery for you, don't hesitate to call a Cobra : they are here for that :)

Ready ?

If every thing is cristal clear and good I think **IT IS TIME** for you to start your first day of pool !

Ready... set... **CODE !**

My_Name

Files to submit: `my_name/my_name.c`

Subject :

The objectiv of that exercise is to write a function that will display your name (**followed by a line break**) in the termainal.

The function has to be prototyped in the following way :

```
void my_name(void);
```

Example :

A call to your function should display these results :

```
my_name();
```

```
$> Timothée
```

How do we do to display a break line again ?

Digits

Files to submit: `digits/digits.c`

Subject :

Write a function, who will dislpay in the terminap all digits in the ascendente order **followed by a break line**

The function has to be prototyped in the following way :

```
void digits(void);
```

Example :

A call to your function with those arguments should display those results :

```
digits();
```

```
$> 0123456789
```

Revalpha

File to submit: `revalpha/revalpha.c`

Subject :

Write a fonction, who will display in the terminal the alphabet in reverse order, one letter on two capital, **followed by a break line**

The function has to be prototyped in the following way :

```
void revalpha(void);
```

Example :

A call to your function with those arguments should display those results :

```
revalpha();
```

```
$> zYxWvUtSrQpOnMlKjIhGfEdCbA
```

Alandroit

File to submit: `alandroit/alandroit.c`

Subject :

Write a fonction, who will display in the terminal the string who would be given as a parameter of the fonction.

The function has to be prototyped in the following way :

```
void alandroit(char *str);
```

Example :

A call to your function with those arguments should display those results :

```
alandroit("EPITECH");
```

```
$> EPITECH
```

```
alandroit("BEEP BOOP");
```

```
$> BEEP BOOP
```

> You have the right at the fonction `strlen` !

Alanver

File to submit: `alanver/alanver.c`

Subject :

Write a function, who will display in the terminal the reverse order of the string given as a parameter of the function.

The function has to be prototyped in the following way :

```
void alanver(char *str);
```

Example :

A call to your function with those arguments should display those results :

```
alanver("EPITECH");
```

```
$> HCETIPE
```

```
alanver("BEEP BOOP");
```

```
$> POOB PEEB
```

```
alanver("bob");
```

```
$> bob
```

Humm... We should certainly write the string from the end, but how to know the end ?

Countoc

File to submit: `countoc/countoc.c`

Subject :

Write a fonction who will return the number of occurrence of the characters **'to_find'** found in the string **'str'**.

The function has to be prototyped in the following way :

```
int countoc(char *str, char to_find);
```

Example :

A call to your function with those arguments should display those results :

```
printf("%d\n", countoc("EPITECH", 'E'));
```

```
$> 2
```

```
printf("%d\n",countoc("##3..##.##....###4#", '.'));
```

```
$> 7
```

Even that `printf` is useable to test at anytime your fonction, this function still not allowed and would be flag by the moulinnette.

Repeat_alpha

File to submit: `repeat_alpha/repeat_alpha.c`

Subject :

For that exercise I let you guess the subject by your self !

You will see it's not that hard !

The function has to be prototyped in the folowing way :

```
void repeat_alpha(char *str);
```

Example :

A call to your function with those arguments should display those results :

```
repeat_alpha("abc");
```

```
$> abbccc
```

```
repeat_alpha("Alex");
```

```
$> Allllllllllleeeeexxxxxxxxxxxxxxxxxxxxxxxx
```

```
repeat_alpha("abacadaba 42!");
```

```
$> abbacccaddddabba 42!
```

If you truely don't see any bond between the exemple and after a long time of reflection, you can call a cobra (don't forget the neighbour rule ;P)

HidenTF

File to submit: `hidentf/hidentf.c`

Subject :

Write a fonction who will return 1 if all letters of s1 are hiden in s2 and 0 if it is the contrary.

The function has to be prototyped in the folowing way :

```
int hidentf(char *s1, char *s2);
```

Example :

A call to your function with those arguments should display those results :

```
printf("%d",hidentf("abc", "abcdefghjikqsdblablabla"));
```

```
$>  1
```

```
printf("%d",hidentf("fgex", "tyf34gdgf;'ektufjhgdgex.;.;rtjynur6"));
```

```
$> 1
```

```
printf("%d",hidentf("azbc", "btarc"));
```

```
$> 0
```

Rot_42

Files submit: `rot_42/rot_42.c`

Subject :

Write a fonction that take an alphabetic string as parameter at first and who would increment each of the letters from 42 than to display it in the terminal **followed by a break line**. If the characters is bigger than the alphabet, we restart from the beggining

The function has to be prototyped in the folowing way :

```
void rot_42(char *str);
```

Example :

A call to your function with those arguments should display those results :

```
rot_42("abc");
```

```
$> qrs
```

```
rot_42("abcdefghijklmnopqrstuvwxyz");
```

```
$> qrstuvwxyzabcdefghijklmnop
```

```
rot_42("Vive le coding club 2022");
```

```
$> Lylu bu setydw sbkr 2022
```

N.b: A letter comeback to 'a' when it is bigger than 'z' and a 'A' when it is bigger than 'Z'

Last_word

files to submit: `last_word/last_word.c`

Subject :

write a fonction who take a string in parameter, and who display the last word of the string, followed by a '\n'.

We call "word" a portion of string who is delimited by spaces and/or tabulation, or at the beggining or end of the string.

The function has to be prototyped in the folowing way :

```
void last_word(char *str);
```

Example :

A call to your function with those arguments should display those results :

```
last_word("EPITECH C'EST PLUTOT COOL QUAND MEME");
```

```
$> MEME
```

```
last_word("Bientot      vous           serez     des     tek    1     ");
```

```
$> 1
```

```
last_word("       lorem,ipsum      ");
```

```
$> lorem,ipsum
```

R_capitalize

Files to submit: `r_capitalize/r_capitalize.c`

Sujet :

write a function reproduising the behavior of the function **"capitalize" of JavaScript**, then display the modified string in the terminal.

THe function will take 2 parameters, the string to modified and an intenger, who will definied the type of the transformation.

- If order equal 0 you should put in capital the first letters of each words and put in lowercase the others.
- If the order is 1, yu should proceed like before, but to put the capitalize at the last one.

If order is not 0 or not 1 you just have to display **'\n'**

The function has to be prototyped in the folowing way :

```
void r_capitalize(char *str, int order);
```

Example :

A call to your function with those arguments should display those results :

```
r_capitalize("EPITECH C'EST PLUTOT COOL QUAND MEME !", 0);
```

```
$> Epitech C'est Plutot Cool Quand Meme !
```

```
r_capitalize("Epitech C'est Plutot Cool Quand Meme !", 1);
```

```
$> epitecH c'esT plutoT cooL quanD memE !
```

It is almost the **END**!

Congrats to arrived here until !

If you did every thing, skip one exercise on two or even if it is 10h10 and that you are reading all the pdf just by curiosity, you are welcome here !

It still only 2 exercise, at the end !

Force and courage for the end young Tek 0 :

Pgcd

Files to submit: `pgcd/pgcd.c`

Subject :

Write a function, who return the biggest commun denominator between 2 numbers.

The function has to be prototyped in the following way :

```
int pgcd(int nb1, int nb2);
```

Example :

A call to your function with those arguments should display those results :

```
printf("%d",pgcd(42, 10))
```

```
$> 2
```

```
printf("%d",pgcd(14, 77))
```

```
$> 7
```

```
printf("%d",pgcd(17, 3))
```

```
$>1
```

Fprime

File to submit: `fprime/fprime.c`

Subject :

Write a function, who take an strict positiv inteneger, and who will display the factorization in prime numbers in the terminal, followed by a '\n'

Factors hve to be displayed in a increasing order and separated by a '*', in a way tha the displayed expression give the good result.

The function has to be prototyped in the following way :

```
void fprime(int nb);
```

Example :

A call to your function with those arguments should display those results :

```
fprime(6)
```

```
$>2*3
```

```
fprime(225225)
```

```
$> 3*3*5*5*7*11*13
```

```
fprime(8333325)
```

```
$> 3*3*5*5*7*11*13*37
```

```
fprime(-1)
```

```
$> -1
```