B1 - Phoenix Bootcamp

B-BOO-101

Day 01

Welcome back

Day 01

repository name: BOO_phoenix_d01_*$ACADEMICYEAR*
repository rights: ramassage-tek
language: C
compilation: gcc *.c (+ our own files containing the `main` function)

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Don't push your `main` and `my_putchar` functions into your delivery directory. Your files will be compiled adding our own `main` and `my_putchar`.

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.

TASK 01 - SHOW_ALPHABET

Delivery: `show_alphabet.c`

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 show_alphabet(void);
```

TASK 02 - SHOW_COMBINATIONS

Delivery: `show_combinations.c`

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 show_combinations(void);
```

```
Terminal
~/B-BOO-101> ./a.out
012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 789
```

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

TASK 03 - SHOW_NUMBER

Delivery: `show_number.c`

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 show_number(int nb);
```

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

TASK 04 - SHOW_STRING

Delivery: `show_string.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 show_string(char const *str);
```

TASK 05 - REVERSE_STRING

Delivery: `reverse_string.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 *reverse_string(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 06 - TO_NUMBER

Delivery: `to_number.c`

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

```
int to_number(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}
```