C1- Unix & C Lab Seminar

pool_c_d04

Day 04

On The Star Road !

Instructions

Before You Go Further

- Turn in directory **pool_c_d04**.
- Respecting the norm takes time, but it is good for you. This way your code will respect the norm from the first written line to the last.
  Read carefully the norm documentation. You should type "alt+i" instead of "tab"
- Do not care about the header. You don't have to add it into your file.
- You shall leave in your directory no other files than those explicitly specified by the exercices. 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.
- You might have to use the NULL keyword. This keyword is located in stddef.h
- **Do not turn-in a main() function unless it has been explicitly asked.**
- **You are only allowed to use the write function to do the following exercices.**
- **Remeber that your exercises should never creash, to prevent that you should always test every cases of functionalities (like empty strings for example)**

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. Do not forget the permissions rights!

Exercise 0

my_init (1pts)

Turn in: pool_c_d04/ex_00/ex_00.c
Prototype: void my_init(int *);

Write a function 'my_init', taking a pointer to an integer as parameter. This function must not return anything but the value of the variable passed in parameter must be set at 42.

Example :
main.c :

```c
#include <stdio.h>

void    my_init(int *);
int     main()
{
  int   i;

  i = 0;
  my_init(&i);
  printf("%d\n", i);
}
```

Terminal

```
~/pool_c_d04> cc *.c -o ex00
~/pool_c_d04> ./ex00
42
~/pool_c_d04>
```

Exercise 1

my_swap (2pts)

Turn in: pool_c_d04/ex_01/ex_01.c
Prototype: void my_swap(int *, int *);

Write a function 'my_swap', taking two pointers to an integer as parameters. This function must not return anything but the value of the two variables passed in parameter must be swapped.

Example :
main.c :

```c
#include <stdio.h>

void    my_swap(int *, int *);
int     main()
{
  int   nb1 = 42;
  int   nb2 = 21;

  printf("%d – %d\n", nb1, nb2);
  my_swap(&nb1, &nb2);
  printf("%d – %d\n", nb1, nb2);
  return (0);
}
```

Terminal

```
~/pool_c_d04> cc *.c -o ex01
~/pool_c_d04> ./ex01
42 - 21
21 - 42
~/pool_c_d04>
```

Exercise 2

my_putstr (3pts)

Turn in: pool_c_d04/ex_02/ex_02.c
Prototype: void my_putstr(char *str);

Write a function 'my_putstr', taking a string as parameter and displaying it on the standard output.

Exercise 3

my_strlen (2pts)

Turn in: pool_c_d04/ex_03/ex_03.c
Prototype: int my_strlen(char *str);

Write a function 'my_strlen', taking a string as parameter and returning its size.

Example :
main.c :

```c
int     my_strlen(char *str);
int     main()
{
  return (my_strlen("Geckos"));
}
```

Terminal

```
~/pool_c_d04> cc *.c -o ex03
~/pool_c_d04> ./ex03
~/pool_c_d04> echo $?
6
~/pool_c_d04>
```

A string always ends with a '\0'

Exercise 4

my_sort_int_tab (3pts)

Turn in: pool_c_d04/ex_04/ex_04.c
Prototype: void my_sort_int_tab(int *tab, int size);

Write a function 'my_sort_int_tab', taking a table made of integers as its first parameter, and the size of the table as a second parameter. The function will sort the table in ascending order.

Exercise 5

my_getnbr (6pts)

Turn in: pool_c_d04/ex_05/ex_05.c
Prototype: int my_getnbr(char *str);

Write a function 'my_getnbr', taking a string as parameter representing a number in base ten ("0123456789") and returning an integer.

Example :
main.c :

```c
#include <stdio.h>

int     my_getnbr(char *str);
int     main()
{
  printf("\"%s\" returns %d\n", "--42", my_getnbr("--42"));
  printf("\"%s\" returns %d\n", "+---+--++---+---+---+-42",
         my_getnbr("+---+--++---+---+---+-42"));
  printf("\"%s\" returns %d\n", "42a43",my_getnbr("42a43"));
  printf("\"%s\" returns %d\n", "-a-a42",my_getnbr("-a-a42"));
  printf("\"%s\" returns %d because this number doesn't fit in an integer.\n",
         "1100000000000000000042",
         my_getnbr("1100000000000000000042"));
  printf("\"%s\" returns %d for the same reason.\n",
         "-1000000000000000000042",
         my_getnbr("-1000000000000000000042"));
  return (0);
}
```

Terminal

```
~/pool_c_d04> cc *.c -o ex05
~/pool_c_d04> ./ex05
"--42" returns 42
"+---+--++---+---+---+-42" returns -42
"42a43" returns 42
"-a-a42" returns 42
"1100000000000000000042" returns 0 because this number doesn't fit in an integer.
"-1000000000000000000042" returns 0 for the same reason.
~/pool_c_d04>
```

Exercise 6

my_show_array (3pts)

Turn in: pool_c_d04/ex_06/ex_06.c
Prototype: void my_show_array(char **tab, int size);

Write a function 'my_show_array', taking an array of string as parameter and displaying the strings from the shortest to the longest one. The function will also take the size of the array as second parameter. If multiple strings have the same length, you will order them from the one having the greater ASCII value to the lesser one. If there are still some equality you must display them following their indexes. For example "you" and "uoy" are perfectly equal, so if tab[0] = "you", you must display "you" before "uoy". Every string must be followed by a '\n'.
Remember that there must always be a null terminated byte in a string. This is how you should test your function :

Example :

```c
void    my_show_array(char **tab, int size);
int     main()
{
  char  tab[3][15] =
    {
      "not prepared !",
      "You",
      "are"
    };
  char  tab2[5][5] =
    {
      "ab",
      "ba",
      "ca",
      "Da",
      "aE"
    };
  char  *f[3] =
    {
      tab[0], tab[1], tab[2]
    };
  char  *f2[5] =
    {
      tab2[0], tab2[1], tab2[2], tab2[3], tab2[4]
    };
  my_show_array(f, 3);
  my_show_array(f2, 5);
  return (0);
}
```

```
~/pool_c_d04> cc *.c -o ex06
~/pool_c_d04> ./ex06
You
are
not prepared !
ca
ab
ba
aE
Da
~/pool_c_d04>
```