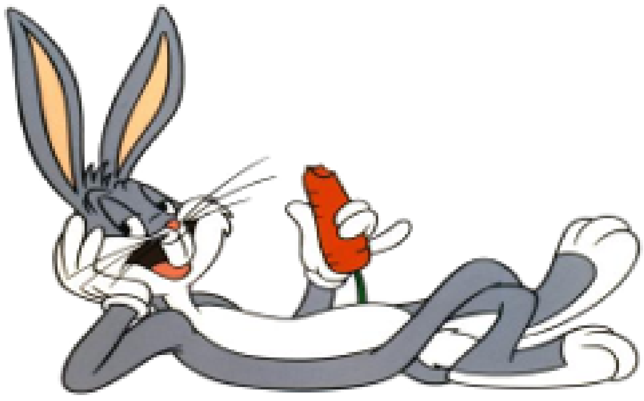

B1- C Pool

B-CPE-042

Day 09

Structures

v1.4

Day 09

Structures

repository name: : CPool_Day09
repository rights: : ramassage-tek
language: : C
group size: : 1
allowed functions: : write, malloc, free

- 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**.
- 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.

All **.c** files from your delivery folder will be collected and compiled with your **libmy**, which is found in **CPool_Day09/lib/my**. For those of you using **.h** files, they must be located in **CPool_Day09/include**.

The Autograder will compile your functions the following way:

```
Terminal
~/B-CPE-042> cd task01
~/B-CPE-042> cc *.c -c -I../include/
~/B-CPE-042> cc *.o ~autograder/main_task01.o -L../lib/my/ -o task01 -lmy
```

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 0

Unit Tests

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-$FUNCTION_NAME.c`, to be stored in the `tests` directory named.
This function must contain all the necessary calls to the task function in order to cover all of the function's possible situations (normal or irregular).

Always check the empty strings and *int*'s special values (0, MIN, MAX)!

Task 1

my_macroABS.h

Write a macro, named **ABS**, that replaces an argument with an absolute value:

```
#define ABS(Value)
```

Delivery: CPool_Day09/task01/my_macroABS.h

Task 2

my.h

Write your **my.h** header file that must contain the prototypes of all the functions found in your **libmy.a**.
Delivery: CPool_Day09/include/my.h

Task 3

my_param_to_tab

Write a function that stores the program's parameters in an array of structures and returnes the address of the array's first cell. All array elements are to be addressed, including **av[0]**.
The function must be prototyped as follows:

```
struct s_stock_par *my_param_to_tab(int ac, char **av);
```

The array's structure is to be allocated, and the last cell will contain **0** in its **str** element, indicating its end.
The structure is defined as follows:

```
struct          s_stock_par
{
        int     size_param;     //parameter's length
        char    *str;           //parameter's address
        char    *copy;          //parameter's copy
        char    **tab;          //returned by my_str_to_wordtab
};
```

Delivery: CPool_Day09/task03/my_param_to_tab.c
Do not submit the **struct s_stock_par** structure; the Autograder will use its own, along with the following typedef:

```
typedef struct s_stock_par t_stock_par;
```

Your function will be tested with **my_show_wordtab**.
As we will not compile **my_show_wordtab.c**, you need to make it work.

Task 4

my_show_tab

Write a function that displays the content of an array created with the previous function, and prototyped as follows:

```
int my_show_tab(struct s_stock_par *par);
```

Do not submit the **struct s_stock_par** structure; the Autograder will use its own, along with the following typedef:

```
typedef struct s_stock_par t_stock_par;
```

For each cell, display one of the following elements per line: parameter, size and word (one per line).
Delivery: CPool_Day09/task04/my_show_tab.c

Your function will be tested with **my_show_wordtab**.
As we will not compile **my_show_wordtab.c**, you need to make it work.

Task 5

get_color

Write a function that returns the color as an **int** by handling its three **RGB** components. The function must be prototyped as follows:

```c
int get_color(unsigned char red, unsigned char green, unsigned char blue);
```

Delivery: CPool_Day09/task05/get_color.c

This task is *only* to be completed with **bit shifts**.

Task 6

swap_endian_color

Write a function changes the endianness of the color and returns it.
The color should be ordered like this: ARGB
The function must be prototyped as follows:

```c
int swap_endian_color(int color);
```

Delivery: CPool_Day09/task06/swap_endian_color.c

This task has to be completed with a **union**.

You will only be working with big and little endian