C1 - Piscine C

C-CPE-042

Day 07

Let’s Code Some Real Programs !

1.1

Day 07

repository name: pool_c_d07
repository rights: ramassage-tek

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).

+ Instructions: Before You Go Further

- Turn in directory: **pool_c_d07**.
- 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"
- 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.
- **Today, you will have to turn-in a main() function.**

+ EXERCISE 1: MY_SHOW_ARGS (4PTS)

Turn in: pool_c_d07/ex_01/
Function allowed for this exercise: printf

Write a **software** 'my_show_args' displaying on the standard output the input parameters followed by a newline. If no parameter is given, you will show only a newline.

Example:

Terminal

```
~/C-CPE-042> cc *.c -o my_show_args
~/C-CPE-042> ./my_show_args Gecko Can Help
Gecko
Can
Help
~/C-CPE-042> ./my_show_args

~/C-CPE-042>
```

+ Exercise 2: my_hello (4pts)

Turn in: pool_c_d07/ex_02/
Function allowed for this exercise: atoi, printf

Write a software 'my_hello' taking a number in parameter. This function must display "Hello" followed by a newline as many time as the number passed in parameter. If nothing is given, or if the number given is negative or null, or if the given parameter is not a number, you should print "Error." followed by a newline.

Example:

Terminal

```
~/C-CPE-042> cc *.c -o my_hello
~/C-CPE-042> ./my_hello 5
Hello
Hello
Hello
Hello
Hello
~/C-CPE-042> ./my_hello
Error.
~/C-CPE-042> ./my_hello a
Error.
~/C-CPE-042>
```

+ EXERCISE 3: MY_WRITE_FILE (4PTS)

Turn in: pool_c_d07/ex_03/
Function allowed for this exercise: read, write, open, close, malloc and free

Write a software 'my_write_file', taking multiple strings in parameter. The first argument will be a file path. You must open it in write mode and erase its content. Then you must write every other parameters of your software in it, each followed by a newline. If the file does not exist, it must be created.

You must close the file at the end of the operation.

If an error occurs, display "Error." followed by a newline on the standard output, and do not create the file. If no line to write if given, then it is an error.

Example:

Terminal

```
∼/C-CPE-042> cc *.c -o my_write_file
∼/C-CPE-042> ./my_write_file gecko.txt hello my friends
∼/C-CPE-042> cat gecko.txt
hello
my
friends
∼/C-CPE-042> ./my_write_file gecko.txt
Error.
∼/C-CPE-042>
```

+ Exercise 4: my_read_file (4pts)

Turn in: pool_c_d07/ex_04/
Function allowed for this exercise: read, write, open, close, malloc and free

Write a software 'my_read_file', taking a string of char in parameter. Your software will be given a file name in parameter. You must open the file in read mode and print its content. If an error occurs, display "Error." followed by a newline on the standard output.

You must close the file at the end of the operation.

Example:

```
Terminal
~/C-CPE-042> cc *.c -o my_read_file
~/C-CPE-042> cat gecko.txt
hello
my
friends
~/C-CPE-042> ./my_read_file gecko.txt
hello
my
friends
~/C-CPE-042>
```

+ Exercise 5: my_connect_four (4pts)

Turn in: pool_c_d07/ex_05/
Function allowed for this exercise: read, write, open, close, malloc and free

Write a software 'my_connect_four'. This software must implement the connect four game. This is a two player game. The goal is to align four identical symbols. Player #1 (P1) will play with 'X' and Player #2 (P2) will play with 'O'. An empty case is fill with '-'.

At each turn, you must dislay a game board with 6 row and 7 columns and ask with a prompt a player to choose in which column he wants to play. An element will always be drop on the choosen column and fall at the bottom of it. If an error occurs, you should ask the player to play again by displaying : "Player #{player number}'s turn :".
Take care, when you prompt "Player #1's turn :", the player should write his move on the same line.
If the game finish with a draw, you should display : "It's a draw !" followed by a new line.

You must use the given function to read the user input :
The str parameter must be allocated.

```
int gecko_read(char *str)
{
    return (read(0, str, 2));
}

void show_game(char **game_array)
{
    int counter_y;

    counter_y = 0;
    printf("| 1 | 2 | 3 | 4 | 5 | 6 | 7 |\n");
    printf("=============================\n");
    while (counter_y < 6)
    {
        printf("| %c | %c | %c | %c | %c | %c | %c |\n",
            game_array[counter_y][0], game_array[counter_y][1],
            game_array[counter_y][2], game_array[counter_y][3],
            game_array[counter_y][4], game_array[counter_y][5],
            game_array[counter_y][6]);
        counter_y++;
    }
    printf("\n");
}
```

Use the previous function for the good formating output.

Example:

```
~/C-CPE-042> cc *.c -o my_connect_four
~/C-CPE-042> ./my_connect_four
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
=============================
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |

Player #1's turn : 4
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
=============================
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | X | - | - | - |

Player #2's turn : 1
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
=============================
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| O | - | - | X | - | - | - |

[...]
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
=============================
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | O | O | - | - | - | - |
| - | O | X | O | - | - | - |
| O | X | X | X | X | - | - |

Congratulations Player #1, you win !
~/C-CPE-042>
```

You should implement a function ‘my_check_winner’, taking the board and the last dropped element as parameter and returning 0, 1 or 2, respectivly no winner, Player #1 win or Player #2 win.