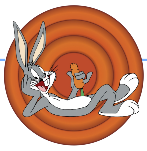

C - Pool - Tek1

Subject Day 05

C Pool Managers
looneytunes@epitech.eu

Contents

Instructions

- The subject may change until one hour before turn-in.
- Respect the norm takes time, but is good for you. This way your code will respect the norm since the first written line.
- Ask yourself if it's relevant to let a `main()` function in your turn-in knowing we will add our own.
- We will compile your files with the command `cc *.c`, adding our `main.c` and our `my_putchar.c` :

```
$> cc *.c ~moulinette/main_ex_01.c ~moulinette/my_putchar.c -o ex01
$> ./ex01
[...]
```

- This is a turn-in directory, of course you will only keep in it your final work revision. No temporary file should stand there!
  You shall leave in your directory no other files than those explicitly specified by the exercises.
  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`. That is why it's in your interest to remove any file that doesn't work.

Hints `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`

- You are only allowed to use the `my_putchar` function to do the following exercises. This function will be provided, so:
  - You shall not have a `my_putchar.c` file in your turn-in directory.
  - The function `my_putchar` shall not be in any of your turned-in files.
- Don't forget to discuss about it in the pool section of the forum !

Unit Tests

- It is highly recommended to test your functions when you are developing them.
- Usually, it is common to create a function named "`main`" (and a dedicated file to host it) to check the functions separately.
- Create a directory named "`tests`".
- Create a function "`int main()`" in a file named "`tests-exercise_name.c`", stored inside the directory "`tests`" previously created.
- According to you, this function must contains all the necessary call to "`exercise_name`" to cover all possible cases (special or regular) of the function.

Indices

```
Here is a partial list of tests:
- Pay attention to the possible values of an int
(0, min and max)
```

Exercise 1 - my_factorielle_it

- Write an iterative function that returns a number. This number is the result of the factorial operation from the number given as a parameter to the function.
- In case of any errors, the function should return 0.
- It will be prototyped as follows:

```
int my_factorielle_it(int nb);
```

- You need to manage if the factorial operation of the number overflows (which is an error).
- Turn-in directory:
  `Piscine_C_J05/my_factorielle_it.c`

Your function must give its answer in less than 2 seconds

Indices

- `my_factorielle_it(0) = 1`
- `If (n < 0) then my_factorielle_it(n) = 0`
- `n power 0 = 1`
- `If (p < 0) then n power p = 0`

Exercise 2 - my_factorielle_rec

- Write a recursive function that returns the factorial operation of the number given as a parameter.
- It will ne prototyped as follows:

```
int my_factorielle_rec(int nb);
```

- Turn-in directory:
  `Piscine_C_J05/my_factorielle_rec.c`

It will give the same answers as the function my_factorielle_it

Exercise 3 - my_power_it

- Write an iterative function that returns the power of a number.
- It will be prototyped as follows:

```c
int my_power_it(int nb, int power);
```

- Turn-in directory:
  `Piscine_C_J05/my_power_it.c`

`Your function must give its answer in less than 2 seconds`

Exercise 4 - my_power_rec

- Write a recursive function that returns the power of a number.
- It will be prototyped as follows:

```
int my_power_rec(int nb, int power);
```

- Turn-in directory:
  `Piscine_C_J05/my_power_rec.c`

It will give the same answers as my_power_it

Exercise 5 - my_square_root

- Write a function that returns the integer square root of a number if it exists and 0 if it's not a whole number.
- It will be prototyped as follows:

```c
int my_square_root(int nb);
```

- Turn-in directory:
  `Piscine_C_J05/my_square_root.c`

Your function must give its answer in less than 2 seconds

Exercise 6 - my_is_prime

- Write a function that returns 1 if the number is prime and 0 if the number is not prime.
- It will be prototyped as follows:

```c
int my_is_prime(int nb);
```

- Turn-in directory:
  `Piscine_C_J05/my_is_prime.c`

Hints `0 and 1 are not prime numbers`

`Your function must give its answer in less than 2 seconds`

Exercise 7 - my_find_prime_sup

- Write a function that returns the prime number immediately higher or equal to the number given as a parameter.
- It will be prototyped as follows:

```
int my_find_prime_sup(int nb)
```

- Turn-in directory:
  `Piscine_C_J05/my_find_prime_sup.c`

`Your function must give its answer in less than 2 seconds`

Exercise 8 - The 8 queens 1

- The aim of the game is to place 8 queens on a chessboard without them being able to run into each other in a single move.
- Technical details:
  - A chessboard is composed of 8x8 squares.
  - A queen can move in rows, column and diagonal.
- Of course, we will use recursion to solve this problem.
- Write a function that returns the number of possibilities of placing 8 queens on a chessboard without them being able to run into each other in a single move.
- It will be prototyped as follows:

```
int my_8r1();
```

- Turn-in directory:
  `Piscine_C_J05/my_8r/my_8r1.c`

`Your function must give its answer in less than 2 seconds.`

Exercise 9 - The 8 queens 2

- Write a function that diplays every possibilities of placing 8 queen on a chessboard without them being able to run into each others in a single move.
- Recursion must be used.
- It will be prototyped as follows:

```
int my_8r2();
```

- Turn-in directory:
  `Piscine_C_J05/my_8r/my_8r2.c`
- The display will be as follows (the results below are false, they only illustrate the display):

```
15346872
13564287
...
```

- There is a line break after the last solution of the 8 queens problem.

`Your function must give its answer in less than 2 seconds`

That's all Folks!