**Quest01**

Remember to git add && git commit && git push each exercise!

We will execute your function with our test(s), please DO NOT PROVIDE ANY TEST(S) in your file

For each exercise, you will have to create a folder and in this folder, you will have additional files that contain your work. Folder names are provided at the beginning of each exercise under submit directory and specific file names for each exercise are also provided at the beginning of each exercise under submit file(s).

**Introduction**

Welcome to the first coding with C quest.

C is a very powerful language, most of all new language are either using C syntax or built on C (Python / Ruby / Javascript / C++ / ...)

C is not easy because you have to handle types and memory but being an expert in those two areas will give you a very powerful advantage.

This quest will lead you to the basic C syntax (variable, if, while-loop, functions and ASCII) You will use your first System Call: write().

If we continue with our comparison on learning how to stand up before being able to walk then run in order to enjoy playing a sport with this quest you will learn to walk! :-)

**My First Compilation**

* Submit directory: ex00
* Submit file: ["my\_first\_compilation.c"]

Last part of coding is to compile but we will start by this part. :)

What is compilation? It transforms a text file (yes a file of code is a text file) into a binary file.

How to compile?

gcc -o my\_first\_compilation my\_first\_compilation.c

-o stands for output, it will be the name of the binary we want to create.

xxxx.c -> C file we want to compile

Let's dive in.

**Step00**

We will use this C file (you can copy paste it). You need to name it: my\_first\_compilation.c

#include <stdio.h>

int main(int ac, char \*\*av) {

printf("my\_first\_compilation.\n");

return 0;

}

**Step01**

Run the compilation command:

gcc -o my\_first\_compilation my\_first\_compilation.c

**Step02**

Execution

./my\_first\_compilation

It should print my\_first\_compilation. :)

(The \n means it will go to a new line.)

**Example 00**

Input:

Output: my\_first\_compilation.

Return Value: nil

*Tip* To test if your exercise(s) is/are correct(s), you can execute the command gandalf in your terminal.

**My First Variable Integer**

* Submit directory: ex01
* Submit file: ["my\_first\_variable\_integer.c"]

First part of coding is to create variable. Let's get started with an integer variable. What does it mean "integer variable?" In most languages you have "types", a good comparaison is a letter is different from a number. In a computer everything is numbers (0 and 1). But we, as human, interact with letter (and words) to make it usuable there are "convention": a letter is a number and one of this table of convertion is: ASCII (you should google man ascii)

Enought talking!

Replace/Complete the following code. Create a variable with (if needed) the right type. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

int main() {

XX = 34;

printf("%d\n", person\_age);

return 0;

}

**Example 00**

Input:

Output: 34

Return Value: nil

**My First Variable Char**

* Submit directory: ex02
* Submit file: ["my\_first\_variable\_char.c"]

The first part of coding is to create a variable. Let's get started with an integer variable. What does an "integer variable” mean? In most languages, you have "types". A good comparison is that a letter is different from a number. In a computer, everything is numbered (0 and 1). But we, as humans, interact with a letter (and words) to make it useable there is "convention": a letter is a number and one of this table of conversion is: ASCII (you should google man ASCII)

Enough talking!

Replace/Complete the following code. Create a variable with (if needed) the right type. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

int main() {

XX = 'c';

printf("%c\n", my\_letter);

return 0;

}

**Example 00**

Input:

Output: c

Return Value: nil

**My First Variable String**

* Submit directory: ex03
* Submit file: ["my\_first\_variable\_string.c"]

What is a string?, a word? How a computer is creating a string? It could be defined by "multiple letter", which is translated to multiple "characters." Is it an array of characters? :-)

Replace/Complete the following code. Create a variable with (if needed) the right type. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

int main() {

XX = "Learning is growing";

printf("%s\n", my\_string);

return 0;

}

**Example 00**

Input:

Output: Learning is growing

Return Value: nil

**My Multiple Variables Multiple Type**

* Submit directory: ex04
* Submit file: ["my\_multiple\_variables\_multiple\_type.c"]

Replace/Complete the following code. Create multiple variables with (if needed) the right type. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

int main() {

XX = 34;

XX = "Luke";

XX = ',';

printf("Hello %s%c I'm %d years old.\n", my\_name, my\_comma, my\_age);

return 0;

}

**Example 00**

Input:

Output: Hello Luke, I'm 34 years old.

Return Value: nil

**My First Incrementation**

* Submit directory: ex05
* Submit file: ["my\_first\_incrementation.c"]

Incrementation and decrementation depending of the language it's either ++ (--) or += 1 (-= 1).

Replace/Complete the following code. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

int main() {

int my\_index = 0;

// replace this comment with an increment

printf("%d\n", my\_index);

// replace this comment with an decrement

// replace this comment with an decrement

printf("%d\n", my\_index);

// replace this comment with an increment

// replace this comment with an increment

// replace this comment with an increment

printf("%d\n", my\_index);

return 0;

}

**Example 00**

Input:

Output: 1

-1

2

Return Value: nil

**My First If Else**

* Submit directory: ex06
* Submit file: ["my\_first\_if\_else.c"]

if statement is linked to else are part of the fundamental of coding. Key is to put our effort to design the best condition.

Replace/Complete the following code. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

int main() {

int nbr = 10;

if (XX) {

printf("nbr is greater than 20\n");

}

else {

printf("nbr is less than 20\n");

}

return 0;

}

**Example 00**

Input:

Output: nbr is less than 20

Return Value: nil

**My First If Multiple Conditions**

* Submit directory: ex07
* Submit file: ["my\_first\_if\_multiple\_conditions.c"]

if statement is linked to else and writing the right condition can be quite complicated :D.

Replace/Complete the following code. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

int main() {

int a = 10;

int b = 9;

int c = 11;

int d = 10;

int y = 9;

int z = 11;

if (XX) {

printf("a is bigger than b AND smaller than c AND equal to d\n");

}

if (XX) {

printf("z OR y are bigger than a\n");

}

return 0;

}

**Example 00**

Input:

Output: a is bigger than b AND smaller than c AND equal to d

z OR y are bigger than a

Return Value: nil

**My First Function**

* Submit directory: ex08
* Submit file: ["my\_first\_function.c"]

Writing syntax of code is the small visible part of the Code Iceberg. Software Architecture (Design Software) are really the deal.

In order to "organize" your code function are the key. Let's dive in!

Replace/Complete the following code. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

// Following XXXXXX will be a function that will print using printf("my\_first\_function\n");

XXXXXX

XXXXXX

XXXXXX

int main() {

my\_first\_function();

return 0;

}

**Example 00**

Input:

Output: my\_first\_function

Return Value: nil

*Tip* (In C) Use void as return type for this exercise.

**My First While**

* Submit directory: ex09
* Submit file: ["my\_first\_while.c"]

Repeating is annoying? what is we could create a program for it? :)

Implemente a while loop to print 100 times "I want to code". (Don't forget to increment the index ;-))

Replace/Complete the following code. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

int main() {

int index = 0;

while (XX) {

printf("I want to code\n");

XX

}

return 0;

}

**Example 00**

Input:

Output: I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

I want to code

Return Value: nil

**My First Param Function**

* Submit directory: ex10
* Submit file: ["my\_first\_param\_function.c"]

Function accepts parameters, let's send an integer to our function and print it!

Implemente a while loop to call a function detonation in...X secondes. Your loop will stop a 0. 10 included, 0 is not. (Don't forget to decrement the index ;-))

Replace/Complete the following code. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

// function will printf("detonation in... %d secondes.\n", seconds\_left);

int main() {

timer = 10;

while (XX) {

detonation\_in(timer);

XX

}

return 0;

}

**Example 00**

Input:

Output: detonation in... 10 secondes.

detonation in... 9 secondes.

detonation in... 8 secondes.

detonation in... 7 secondes.

detonation in... 6 secondes.

detonation in... 5 secondes.

detonation in... 4 secondes.

detonation in... 3 secondes.

detonation in... 2 secondes.

detonation in... 1 secondes.

Return Value: nil

*Tip* (In C) Each parameter has its type associated inside the "prototype" of the function

**My First Return Function**

* Submit directory: ex11
* Submit file: ["my\_first\_return\_function.c"]

Function returns a value, let's print it!

Implemente a function which return a number (7)

Replace/Complete the following code. (XX is what you need to replace)

**Function prototype** (c)

#include <stdio.h>

// function my\_get\_seven() will return 7

int main() {

printf("%d\n", my\_get\_seven());

return 0;

}

**Example 00**

Input:

Output: 7

Return Value: nil

*Tip* (In C) Return type is part of the "prototype" of the function

**My Is Negative**

* Submit directory: ex12
* Submit file: ["my\_is\_negative.c"]

Let's get starting with some if-else statement!

Create a my\_is\_negative function. This function my\_is\_negative returns 1 or 0 depending on the integer's sign entered as a parameter. If n is negative, return 0. If n is positive or 0, return 1.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_is\_negative

\*\*

\*\* @param {int} param\_1

\*\*

\*\* @return {int}

\*\*

\*/

int my\_is\_negative(int param\_1)

{

}

*Tip* (In C) Your script will look like something close to this:

int my\_is\_negative(int nbr) {

if (XXXXX) {

return XXX;

}

else {

return XXX;

}

}

printf("-> %d\n", my\_is\_negative(-1));

printf("-> %d\n", my\_is\_negative(1));

printf("-> %d\n", my\_is\_negative(0));

// printf("-> %d\n", my\_is\_negative(1337));

// REMEMBER WHEN YOU ARE FINISHED TO COMMENT ALL CALL TO YOUR

// FUNCTION my\_is\_negative function

// OTHERWISE IT WILL FAIL THE AUTOMATIC TEST SYSTEM

//

// <- yes this a way to comment your code

**My Abs**

* Submit directory: ex13
* Submit file: ["my\_abs.c"]

Create a my\_abs function.

Reproduce behavior of an abs() function. It returns always the positive value of a number.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_abs

\*\*

\*\* @param {int} param\_1

\*\*

\*\* @return {int}

\*\*

\*/

int my\_abs(int param\_1)

{

}

**Example 00**

Input: -30

Output:

Return Value: 30

**Example 01**

Input: 30

Output:

Return Value: 30

**Example 02**

Input: 0

Output:

Return Value: 0

**My Isalpha**

* Submit directory: ex14
* Submit file: ["my\_isalpha.c"]

Create a my\_isalpha function.

Reproduce the behavior of isalpha() function. It returns 1 if the character sent as argument is a letter (A to Z or a to z). It returns 0 otherwise.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_isalpha

\*\*

\*\* @param {char} param\_1

\*\*

\*\* @return {int}

\*\*

\*/

int my\_isalpha(char param\_1)

{

}

**Example 00**

Input: "a"

Output:

Return Value: 1

**Example 01**

Input: " "

Output:

Return Value: 0

**Example 02**

Input: "0"

Output:

Return Value: 0

*Tips* (In C) man ascii (In C) man isalpha

**My Isdigit**

* Submit directory: ex15
* Submit file: ["my\_isdigit.c"]

Create a my\_isdigit function.

Reproduce the behavior of isdigit() function. It returns 1 if the character sent as argument is a digit (0 to 9). It returns 0 otherwise.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_isdigit

\*\*

\*\* @param {char} param\_1

\*\*

\*\* @return {int}

\*\*

\*/

int my\_isdigit(char param\_1)

{

}

**Example 00**

Input: "a"

Output:

Return Value: 0

**Example 01**

Input: " "

Output:

Return Value: 0

**Example 02**

Input: "0"

Output:

Return Value: 1

*Tips* (In C) man ascii (In C) man isdigit

**My Islower**

* Submit directory: ex16
* Submit file: ["my\_islower.c"]

Create a my\_islower function.

Reproduce the behavior of islower() function. It returns 1 if the character sent as argument is a lower letter (a to z). It returns 0 otherwise.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_islower

\*\*

\*\* @param {char} param\_1

\*\*

\*\* @return {int}

\*\*

\*/

int my\_islower(char param\_1)

{

}

**Example 00**

Input: "a"

Output:

Return Value: 1

**Example 01**

Input: "A"

Output:

Return Value: 0

**Example 02**

Input: "0"

Output:

Return Value: 0

*Tips* (In C) man ascii (In C) man islower

**My Isupper**

* Submit directory: ex17
* Submit file: ["my\_isupper.c"]

Create a my\_isupper function.

Reproduce the behavior of isupper() function. It returns 1 if the character sent as argument is a upper-case letter (A to Z). It returns 0 otherwise.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_isupper

\*\*

\*\* @param {char} param\_1

\*\*

\*\* @return {int}

\*\*

\*/

int my\_isupper(char param\_1)

{

}

**Example 00**

Input: "a"

Output:

Return Value: 0

**Example 01**

Input: "A"

Output:

Return Value: 1

**Example 02**

Input: "0"

Output:

Return Value: 0

*Tips* (In C) man ascii (In C) man isupper

**My Isspace**

* Submit directory: ex18
* Submit file: ["my\_isspace.c"]

Create a my\_isspace function.

Reproduce the behavior of isspace() function. It returns 1 if the character sent as argument is a space (man isspace). It returns 0 otherwise.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_isspace

\*\*

\*\* @param {char} param\_1

\*\*

\*\* @return {int}

\*\*

\*/

int my\_isspace(char param\_1)

{

}

**Example 00**

Input: "a"

Output:

Return Value: 0

**Example 01**

Input: "A"

Output:

Return Value: 0

**Example 02**

Input: " "

Output:

Return Value: 1

*Tips* (In C) man ascii (In C) man isspace

**My Print Alphabet**

* Submit directory: ex19
* Submit file: ["my\_print\_alphabet.c"]

Create a function that displays the alphabet in lowercase, on a single line, by ascending order, starting from the letter a. It will be follow by a \n (newline character)

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_print\_alphabet

\*\*

\*\*

\*\* @return {void}

\*\*

\*/

void my\_print\_alphabet()

{

}

**Example 00**

Input:

Output: abcdefghijklmnopqrstuvwxyz

Return Value: nil

*Tip* (In C) In order to print here is a function you can copy and paste:

void my\_putchar(char c) {

write(1, &c, 1);

}

and to use it:

my\_putchar("a");

**My Print Reverse Alphabet**

* Submit directory: ex20
* Submit file: ["my\_print\_reverse\_alphabet.c"]

Create a function that displays the alphabet in lowercase, on a single line, by descending order, starting from the letter z. It will be follow by a \n (newline character)

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_print\_reverse\_alphabet

\*\*

\*\*

\*\* @return {void}

\*\*

\*/

void my\_print\_reverse\_alphabet()

{

}

**Example 00**

Input:

Output: zyxwvutsrqponmlkjihgfedcba

Return Value: nil

*Tip* (In C) In order to print here is a function you can copy and paste:

void my\_putchar(char c) {

write(1, &c, 1);

}

and to use it:

my\_putchar("a");