**Quest02**

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**

Pointers!!

In our life as developer there is a before and after in our life with pointers :D There is no better way to learn pointers... than doing it.

What is a pointer?

A variable which contain the address of another variable.

^o) Let dive... in?

Objective of this quest is to discover pointers and starting to have some experience with pointers. We will also do some loop. :)

**My Initializer**

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

Create a function that takes a pointer to integer as a parameter, and sets the value to 0.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_initializer

\*\*

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

\*\*

\*\* @return {void}

\*\*

\*/

void my\_initializer(int\* param\_1)

{

}

**Example 00 (In C)**

int main() {

int variable\_a = 12;

printf("%d\n", variable\_a); // will print 12

my\_initializer(&variable\_a);

printf("%d\n", variable\_a); // will print 0

return 0;

}

**My Swap**

* Submit directory: ex01
* Submit file: ["my\_swap.c"]

Let's switch the content of parameter A and parameter B. :-)

Create a function that swaps the value of two integers whose addresses are entered as parameters.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_swap

\*\*

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

\*\* @param {int\*} param\_2

\*\*

\*\* @return {void}

\*\*

\*/

void my\_swap(int\* param\_1, int\* param\_2)

{

}

**Example 00 (In C)**

int main() {

int variable\_a = 12;

int variable\_b = 21;

printf("A - %d \*\* B - %d\n", variable\_a, variable\_b); // will print A - 12 \*\* B - 21

my\_swap(&variable\_a, &variable\_b);

printf("A - %d \*\* B - %d\n", variable\_a, variable\_b); // will print A - 21 \*\* B - 12

return 0;

}

*Tip* (In C) Pointer is a key element here. Remember a pointer is a variable that contains the address of another variable.

**My Strlen**

* Submit directory: ex02
* Submit file: ["my\_strlen.c"]

Reproduce the behavior of the function strlen. The strlen() function computes the length of the string s.

The strlen() function returns the number of characters. **C Prototype**:

size\_t my\_strlen(const char \*s);

**Example 00**:

Input: "abc"

Output: 3

**Example 01**:

Input: "RaInB0w d4Sh!"

Output: 13

**Example 02**:

Input: "ThE C4k3 Is a L|3"

Output: 17

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_strlen

\*\*

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

\*\*

\*\* @return {int}

\*\*

\*/

int my\_strlen(char\* param\_1)

{

}

**My Putstr**

* Submit directory: ex03
* Submit file: ["my\_putstr.c"]

Create a function that displays a string of characters on the standard output. The address of the string's first character is in the pointer entered as parameter in the function.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_putstr

\*\*

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

\*\*

\*\* @return {void}

\*\*

\*/

void my\_putstr(char\* param\_1)

{

}

**Example 00**

Input: "abc"

Output: abc

Return Value: nil

**Example 01**

Input: "abcdelele dzp ll 0"

Output: abcdelele dzp ll 0

Return Value: nil

**Example 02**

Input: ""

Output:

Return Value: nil

*Tips* (In C) Remember \0 is the End Of String (In C) To print a character you can use my\_putchar

int my\_putchar(char c) {

return write(1, &c, 1);

}

**My Add**

* Submit directory: ex04
* Submit file: ["my\_add.c"]

Create a my\_add function which takes 2 parameters (nbr1 and nbr2) and returns a value. This value is the result of the addition of nbr1 and nbr2 parameters.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_add

\*\*

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

\*\* @param {int} param\_2

\*\*

\*\* @return {int}

\*\*

\*/

int my\_add(int param\_1, int param\_2)

{

}

**Example 00**

Input: 0 && 1

Output:

Return Value: 1

**Example 01**

Input: 10 && 10

Output:

Return Value: 20

**Example 02**

Input: -10 && 10

Output:

Return Value: 0

**My Sub**

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

Create a my\_sub function which takes 2 parameters (nbr1 and nbr2) and returns a value. This value is the result of the subtraction of nbr1 and nbr2 parameters.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_sub

\*\*

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

\*\* @param {int} param\_2

\*\*

\*\* @return {int}

\*\*

\*/

int my\_sub(int param\_1, int param\_2)

{

}

**Example 00**

Input: 0 && 1

Output:

Return Value: -1

**Example 01**

Input: 10 && 10

Output:

Return Value: 0

**Example 02**

Input: -10 && 10

Output:

Return Value: -20

**My Mult**

* Submit directory: ex06
* Submit file: ["my\_mult.c"]

Create a my\_mult function which takes 2 parameters (nbr1 and nbr2) and returns a value. This value is the result of the multiplication of nbr1 and nbr2 parameters.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_mult

\*\*

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

\*\* @param {int} param\_2

\*\*

\*\* @return {int}

\*\*

\*/

int my\_mult(int param\_1, int param\_2)

{

}

**Example 00**

Input: 0 && 1

Output:

Return Value: 0

**Example 01**

Input: 10 && 10

Output:

Return Value: 100

**Example 02**

Input: -10 && 10

Output:

Return Value: -100

**My String Formatting**

* Submit directory: ex07
* Submit file: ["my\_string\_formatting.c"]

Create a my\_string\_formatting function which takes 3 parameters (firstname, lastname and age) and prints a string composed value.

Formatting should be: "Hello, my name is FIRSTNAME LASTNAME, I'm AGE."

There is a newline added.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_string\_formatting

\*\*

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

\*\* @param {char\*} param\_2

\*\* @param {int} param\_3

\*\*

\*\* @return {void}

\*\*

\*/

void my\_string\_formatting(char\* param\_1, char\* param\_2, int param\_3)

{

}

**Example 00**

Input: "john" && "doe" && 37

Output: Hello, my name is john doe, I'm 37.

Return Value: nil

**Example 01**

Input: "Baby" && "Yoda" && 50

Output: Hello, my name is Baby Yoda, I'm 50.

Return Value: nil

**Example 02**

Input: "Marie" && "Curie" && 26

Output: Hello, my name is Marie Curie, I'm 26.

Return Value: nil

*Tip* You should use Google to learn about String interpolation :-)

**My String Index**

* Submit directory: ex08
* Submit file: ["my\_string\_index.c"]

Create a my\_string\_index function which takes 2 parameters (haystack and needle) and locates the first occurrence of the character needle in the string haystack and returns the position.

You can this function as: is there a L (character) in my string "helLo".

Objective is to build a loop and having a if statement when it matches the right character and returns its position.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_string\_index

\*\*

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

\*\* @param {char} param\_2

\*\*

\*\* @return {int}

\*\*

\*/

int my\_string\_index(char\* param\_1, char param\_2)

{

}

**Example 00**

Input: "hello" && "l"

Output:

Return Value: 2

**Example 01**

Input: "aaaaa" && "b"

Output:

Return Value: -1

**My Upcase**

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

Create a my\_upcase function. Which takes a string as parameter and returns the uppercase version of it.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_upcase

\*\*

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

\*\*

\*\* @return {char\*}

\*\*

\*/

char\* my\_upcase(char\* param\_1)

{

}

**Example 00**

Input: "aBc"

Output:

Return Value: "ABC"

**Example 01**

Input: ""

Output:

Return Value: ""

*Tips* Google upcase string YOURCODINGLANGUAGE (In C)

/\*

Example of main

\*/

int main() {

char \*my\_str = strdup("AbcE Fgef1");

printf("RANDOM CASE -> %s\n", my\_str);

printf("UPCASE -> %s\n", my\_upcase(my\_str));

return 0;

}

**My Downcase**

* Submit directory: ex10
* Submit file: ["my\_downcase.c"]

Create a my\_downcase function. Which takes a string as parameter and returns the lowercase version of it.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_downcase

\*\*

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

\*\*

\*\* @return {char\*}

\*\*

\*/

char\* my\_downcase(char\* param\_1)

{

}

**Example 00**

Input: "aBc"

Output:

Return Value: "abc"

**Example 01**

Input: ""

Output:

Return Value: ""

*Tips* Google downcase string YOURCODINGLANGUAGE (In C)

/\*

Example of main

\*/

int main() {

char \*my\_str = strdup("AbcE Fgef1");

printf("RANDOM CASE -> %s\n", my\_str);

printf("DOWNCASE -> %s\n", my\_downcase(my\_str));

return 0;

}