**Quest03**

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**

Let's practice.

We will proceed to rebuild some of the C library function to understand how they work.

Objective: More pointers, more loops, more string and a beginning of struct. :-)

Complexity will be one loop with multiple variables. strstr is specially you will touch the one loop with one nested loop. :-)

Last notion is struct, it's a big box of variables :-)

**Reverse String**

* Submit directory: ex00
* Submit file: ["reverse\_string.c"]

Write a program that takes a string as argument returns its reverses.

Your algorithm must be IN PLACE. (What is in place?) An in-place algorithm is an algorithm which transforms input using no auxiliary data structure.

**Example 00**:

Input: "Hello"

Output: "olleH"

**Example 01**:

Input: "LoL"

Output: "LoL"

**Example 02**:

Input: "Nothing Else Matters"

Output: "srettaM eslE gnihtoN"

**Example 03**:

Input: ""

Output: ""

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- reverse\_string

\*\*

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

\*\*

\*\* @return {char\*}

\*\*

\*/

char\* reverse\_string(char\* param\_1)

{

}

*Tip* (In C)

/\*

Example of main

\*/

int main() {

char \*my\_str = strdup("Hello");

printf("Before reverse -> %s\n", my\_str);

printf("Reverse -> %s\n", reverse\_string(my\_str));

return 0;

}

**My Strcmp**

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

Reproduce the behavior of the function strcmp. The strcmp() function compares string1 with string2 to see if there are equals.

**Tip** Return value is connected to ASCII values ;-)

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_strcmp

\*\*

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

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

\*\*

\*\* @return {int}

\*\*

\*/

int my\_strcmp(char\* param\_1, char\* param\_2)

{

}

**Example 00**

Input: "abc" && "bd"

Output:

Return Value: -1

**Example 01**

Input: "bd" && "abc"

Output:

Return Value: 1

**Example 02**

Input: "abc" && "abc"

Output:

Return Value: 0

*Tip* (In C)

/\*

Example of main

\*/

int main() {

char \*s1 = "Hello";

char \*s2 = "Hello";

printf("my\_strcmp -> %d\n", my\_strcmp(s1, s2));

return 0

}

**My Strcpy**

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

Reproduce the behavior of the function strcpy. The stpcpy() and strncpy() functions copy the string src to dst.

The strcpy() and strncpy() functions return dst.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_strcpy

\*\*

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

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

\*\*

\*\* @return {char\*}

\*\*

\*/

char\* my\_strcpy(char\* param\_1, char\* param\_2)

{

}

**Example 00**

Input: "" && "abc"

Output:

Return Value: "abc"

**Example 01**

Input: "" && "RaInB0w d4Sh! "

Output:

Return Value: "RaInB0w d4Sh! "

**Example 02**

Input: "" && ""

Output:

Return Value: ""

*Tip* (In C)

/\*

Example of main

\*/

#include <stdio.h>

int main() {

char dst[100] = {0};

char \*src = "Hello";

printf("my\_strcpy -> %s\n", my\_strcpy(dst, src));

return 0;

}

**My Strncpy**

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

Reproduce the behavior of the function strncpy.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_strncpy

\*\*

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

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

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

\*\*

\*\* @return {char\*}

\*\*

\*/

char\* my\_strncpy(char\* param\_1, char\* param\_2, int param\_3)

{

}

**Example 00**

Input: "" && "abc" && 2

Output:

Return Value: "ab"

**Example 01**

Input: "" && "RaInB0w d4Sh! " && 6

Output:

Return Value: "RaInB0"

**Example 02**

Input: "" && "Hello World" && 0

Output:

Return Value: ""

**My Strchr**

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

Searches for the first occurrence of the character *c* (an unsigned char) in the string pointed to by the argument *str*. The terminating null character is considered to be part of the string. Returns a pointer pointing to the last matching character, or null if no match was found.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_strchr

\*\*

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

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

\*\*

\*\* @return {char\*}

\*\*

\*/

char\* my\_strchr(char\* param\_1, char param\_2)

{

}

**Example 00**

Input: "abcabc" && "b"

Output:

Return Value: "bcabc"

**Example 01**

Input: "121212" && "2"

Output:

Return Value: "21212"

**Example 02**

Input: "abc" && "d"

Output:

Return Value: nil

*Tip* (In C) nil in C is NULL

**My Strrchr**

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

Searches for the last occurrence of the character *c* (an unsigned char) in the string pointed to by the argument *str*. The terminating null character is considered to be part of the string. Returns a pointer pointing to the last matching character, or null if no match was found.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_strrchr

\*\*

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

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

\*\*

\*\* @return {char\*}

\*\*

\*/

char\* my\_strrchr(char\* param\_1, char param\_2)

{

}

**Example 00**

Input: "abcabc" && "b"

Output:

Return Value: "bc"

**Example 01**

Input: "121212" && "2"

Output:

Return Value: "2"

**Example 02**

Input: "abc" && "d"

Output:

Return Value: nil

*Tip* (In C) nil in C is NULL

**My Strstr**

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

Implement [strStr()](http://www.cplusplus.com/reference/cstring/strstr/).

Returns a pointer to the first occurrence of needle in haystack, or **NULL** if needle is not part of haystack.

**Clarification:**

What should we return when needle is an empty string? This is a great question to ask during an interview.

For the purpose of this problem, we will return haystack when needle is an empty string.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_strstr

\*\*

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

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

\*\*

\*\* @return {char\*}

\*\*

\*/

char\* my\_strstr(char\* param\_1, char\* param\_2)

{

}

**Example 00**

Input: "hello" && "ll"

Output:

Return Value: "llo"

**Example 01**

Input: "aaaaa" && "bba"

Output:

Return Value: nil

**Example 02**

Input: "" && ""

Output:

Return Value: ""

*Tip* (In C) pseudo-code:

foreach letter\_s1 in s1

foreach letter\_s2 in s2

if letter\_s1 != letter\_s2

break

if reach end of s2

return &s1[index]

return NULL

**My First Struct**

* Submit directory: ex07
* Submit file: ["my\_first\_struct.c"]

Structure in C are similar to the concept of package.

You can have multiple object inside your package and with a struct you can have multiple variable in it.

A struct is a type.

You are defining a struct of type struct s\_box which will contains variables.

example:

struct s\_coordinate {

int x;

int y;

}

int main() {

struct s\_coordinate coord;

coord.x = 0;

coord.y = 0;

return 0;

}

A struct has a special behavior if it's a pointer on a struct or not.

If it's a pointer on a struct, in order to reach the variable you will use -> example:

struct s\_coordinate {

int x;

int y;

}

int main() {

struct s\_coordinate coord;

struct s\_coordinate\* ptr\_on\_coord = &coord;

ptr\_on\_coord->x = 0;

ptr\_on\_coord->y = 0;

return 0;

}

Let's dive in. :-)

In this exercise you will receive a struct integer\_array. You have to print its size and its content following this format: size\narray[0]\narray[1]... Iterate throught the variable size to iterate throught the struct.

If the size is 0, just print 0\n.

You can use printf() to print.

**Function prototype** (c)

/\*

\*\*

\*\* QWASAR.IO -- my\_first\_struct

\*\*

\*\* @param {integer\_array\*} param\_1

\*\*

\*\* @return {void}

\*\*

\*/

#ifndef STRUCT\_INTEGER\_ARRAY

#define STRUCT\_INTEGER\_ARRAY

typedef struct s\_integer\_array

{

int size;

int\* array;

} integer\_array;

#endif

void my\_first\_struct(integer\_array\* param\_1)

{

}

**Example 00**

Input: [0]

Output: 1

0

Return Value: nil

**Example 01**

Input: [1, 2, 3]

Output: 3

1

2

3

Return Value: nil

**Example 02**

Input: [10, 2, 3, 3, 0, -1]

Output: 6

10

2

3

3

0

-1

Return Value: nil

**My Is Sort**

* Submit directory: ex08
* Submit file: ["my\_is\_sort.c"]

Let's create a function which will tell us if an array is sorted or not. What is sorted? :-)

Write a function that takes an integer array as a parameter (input) and returns a boolean (true/false).

Your function should return true if the integer array is sorted in either ASC(ascending) or DESC(descending) order. Your function should return false if the integer array is not sorted.

Numbers will be from -2\_000\_000 to 2\_000\_000 Array might have duplicate(s).

**Function prototype** (c)

/\*

\*\*

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

\*\*

\*\* @param {integer\_array\*} param\_1

\*\*

\*\* @return {bool}

\*\*

\*/

#include <stdbool.h>

#ifndef STRUCT\_INTEGER\_ARRAY

#define STRUCT\_INTEGER\_ARRAY

typedef struct s\_integer\_array

{

int size;

int\* array;

} integer\_array;

#endif

bool my\_is\_sort(integer\_array\* param\_1)

{

}

**Example 00**

Input: [1, 1, 2]

Output:

Return Value: true

**Example 01**

Input: [2, 1, -1]

Output:

Return Value: true

**Example 02**

Input: [4, 7, 0, 3]

Output:

Return Value: false

**Example 03**

Input: []

Output:

Return Value: true

*Tips* (In C) In C, we have defined boolean on char so you can use the type boolean :) (In C) Curious about the integer\_array type?

typedef struct s\_integer\_array {

int size;

int\* array;

} integer\_array;

integer\_array\_variable->size will give you the size of the array

integer\_array\_variable->array will give you the access to the array

integer\_array\_variable->array[0] => is the first cell of the array

Please do not define the struct in your code when sending to gandalf.

Google: what is a Boolean Google: sort ascending