



# Quest04

Subject

1 Solution

Additional Resources  
(5)

## Quest04

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)**.

Quest04	My Range
Submit directory	ex00
Submit file	my_range.c

### Description

### Control Center



Group formation



Progress



Submitted



Test review



**Finished: approved**



[Go To DoCode](#)



Access:

READ

WRITE

[Go To Gitea](#)

[Keep Working On This Solution](#)

**Also working on  
the project**

Create a function `my_range` which returns a malloc'd array of `integers` . This `integer` array should contain all values between `min` and `max` .

Min included - max excluded.

If `min` value is greater or equal to `max` 's value, a null pointer should be returned.

Don't worry about "free", it will be done in our `main()`.

## Function prototype (c)

```
/*
**
** QWASAR.IO -- my_range
**
** @param {int} param_1
** @param {int} param_2
**
** @return {int*}
**
*/

int* my_range(int param_1, int param_2)
{

}
```

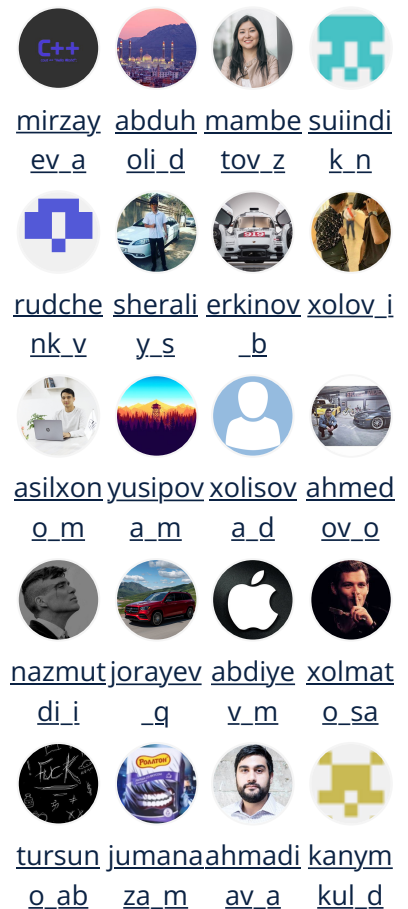
### Example 00

Input: 1 && 4  
Output:  
Return Value: [1, 2, 3]

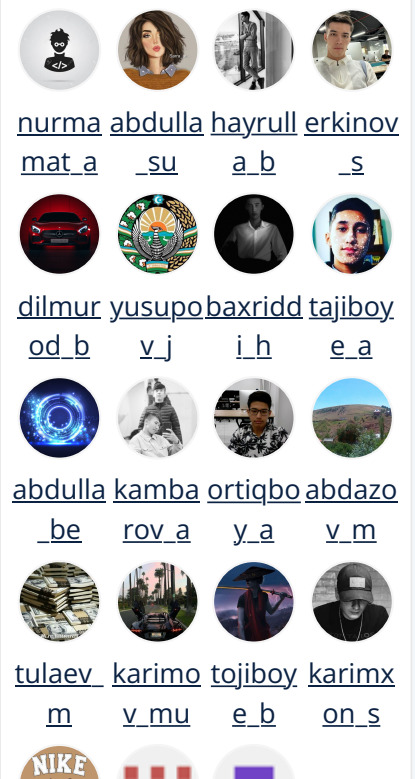
### Example 01

Input: 7 && 10  
Output:  
Return Value: [7, 8, 9]

### Example 02



## Just finished



Input: 10 && 11  
Output:  
Return Value: [10]

Quest04	My Strdup
Submit directory	ex01
Submit file	my_strdup.c

## Description

Let's allocate a string (or array of characters).




We have this: "abc" and we want a copy in a new part of memory that you will have to `malloc` .

(Reproduce the behavior of strdup from `man strdup` )

## Function prototype (c)

```
/*  
**  
** QWASAR.IO -- my_strdup  
**  
** @param {char*} param_1  
**  
** @return {char*}  
**  
**/  
  
char* my_strdup(char* param_1)  
{  
  
}
```

Example 00

    
[sayfulla](#) [belus](#) [yerlanu](#)  
[\\_k](#) [!\\_a](#)

Type  
Project

Group  
Size

1  
Participant

Review  
system

Test Review (Gandalf)

Difficult  
y

Initiation

Averag  
e

1  
Week

duratio  
n

## Project's Metadata

Project

id: 34

name: quest04

visible: True

Input: "abc"  
Output:  
Return Value: "abc"

#### Example 01

Input: "RaInB0"  
Output:  
Return Value: "RaInB0"

#### Example 02

Input: ""  
Output:  
Return Value: ""

#### Tip

(In C)

Don't worry about `free`, it will be done in our `main()`.

---

Quest04	My Print Words Array
Submit directory	ex02
Submit file	my_print_words_array.c

### Description

["Hello", "World", "."]

Create a function that displays the content of an array of strings.

One word per line.

Each word will be followed by a newline, including the last one.

(You can't use `printf`, time to reuse your `my_putstr` function ;-))

## Function prototype (c)

```
/*
**
** QWASAR.IO -- my_print_words_array
**
** @param {string_array*} param_1
**
** @return {void}
**
*/
#ifdef STRUCT_STRING_ARRAY
#define STRUCT_STRING_ARRAY
typedef struct s_string_array
{
    int size;
    char** array;
} string_array;
#endif

void my_print_words_array(string_array*
param_1)
{

}
```

### Example 00

Input: ["abc", "def", "gh"]  
Output: abc  
def  
gh  
  
Return Value: nil

### Example 01

Input: ["123"]  
Output: 123  
  
Return Value: nil

## Example 02

Input: [ "", "", "hello"]

Output:

hello

Return Value: nil

---

Quest04	My Count On It
Submit directory	ex03
Submit file	my_count_on_it.c

## Description

Count the size of each elements in an array.

Create a function `my_count_on_it` , which receives a string array as parameter and returns an array with the length of each strings.

## Function prototype (c)

```

/*
**
** QWASAR.IO -- my_count_on_it
**
** @param {string_array*} param_1
**
** @return {integer_array*}
**
*/
#ifndef STRUCT_STRING_ARRAY
#define STRUCT_STRING_ARRAY
typedef struct s_string_array
{
    int size;
    char** array;
} string_array;
#endif

#ifndef STRUCT_INTEGER_ARRAY
#define STRUCT_INTEGER_ARRAY
typedef struct s_integer_array
{
    int size;
    int* array;
} integer_array;
#endif

integer_array*
my_count_on_it(string_array* param_1)
{

}

```

#### Example 00

Input: ["This", "is", "the", "way"]  
Output:  
Return Value: [4, 2, 3, 3]

#### Example 01

Input: ["aBc", "AbcE Fgef1"]

Output:

Return Value: [3, 10]

#### Example 02

Input: ["aBc"]

Output:

Return Value: [3]

#### Tips

Google while YOURCODINGLANGUAGE

Google for YOURCODINGLANGUAGE

Google array.length YOURCODINGLANGUAGE

(In C)

Yes, you have to allocate for the struct AND then to allocate for the array inside it. :-)

(In C)

Don't forget to set the size. ;-)

---

Quest04	My Join
Submit directory	ex04
Submit file	my_join.c

#### Description

Create a function that join an array of strings on a separator characters.

```
["Hello", "World", "!"] && ' '  
=> "Hello World !"
```

You will receive two parameters, an array with all the string and a separator.



The function returns a string where all the string from array are joined with the separator.

### Function prototype (c)

```
/*
**
** QWASAR.IO -- my_join
**
** @param {string_array*} param_1
** @param {char*} param_2
**
** @return {char*}
**
*/
#ifndef STRUCT_STRING_ARRAY
#define STRUCT_STRING_ARRAY
typedef struct s_string_array
{
    int size;
    char** array;
} string_array;
#endif

char* my_join(string_array* param_1,
char* param_2)
{

}
```

#### Example 00

```
Input: ["abc", "def", "gh", "!"] && "-"
Output:
Return Value: "abc-def-gh-!"
```

#### Example 01

```
Input: ["abc", "def", "gh", "!"] &&
"blah"
Output:
Return Value: "abcbahdefbahghbah!"
```

#### Example 02

```
Input: ["abc", "def", "gh", "!"] && ""
Output:
Return Value: "abcdefgh!"
```

#### Example 03

```
Input: [] && " "
Output:
Return Value: nil
```

---

Quest04	My Spaceship
Submit directory	ex05
Submit file	my_spaceship.c

### Description

You have been recently been hired by SpacePro, a new rocket manufacturer, and have been tasked with designing a simulator for their spaceships. This simulator creates a virtual "space" and keeps track of the basic movements and direction of a given ship. Your job is to keep track of where the ship is and it's orientation relative to its starting point.

### Instructions

Your ship simulator must take in a string of letters that represent a planned flight path for a given rocket ship.

In a ship's flight path there are only 3 valid options for movement; R for turning right, L for turning left and A for advancing.

If, for example, you receive "RRALAA" as your flight path, you should interpret it as the following:

Turn right, turn right, advance, turn left, advance, advance

Once given this initial flight path, your program must return the x,y coordinates of a ship's final destination as well as it's orientation relative to the starting point.

Orientation is represented as left, right, up or down.

Space is infinite, so the x,y coordinates you return could be placed on a seemingly infinite grid and can be negative or positive values.

So let's say an upward facing rocket ship leaves its starting point of 0,0 and is given the flight path of "RRALAA", it's final location will be 2,-1 and it will be facing right.

## Your Job

You must create a function that takes in a flight path of a rocket ship as a string of letters and returns the following format:

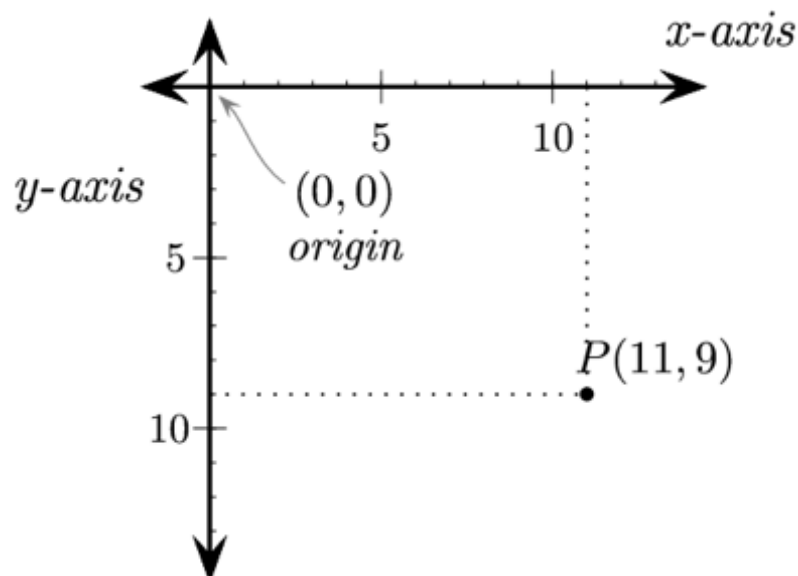
"{x: X, y: Y, direction: 'DIRECTION'}"

X,Y represent the ending coordinates of your ship and direction represents its final direction.

## Notes

Function my\_spaceship returns a *STRING*.

We are using Computer Graphics Coordinate System



All spaceships will start at 0,0 and will face up  
Moving left/right will influence X and moving up/down will influence Y

### Function prototype (c)

```
/*  
**  
** QWASAR.IO -- my_spaceship  
**  
** @param {char*} param_1  
**  
** @return {char*}  
**  
*/  
  
char* my_spaceship(char* param_1)  
{  
  
}
```

#### Example 00

```
Input: "RAALALL"  
Output:  
Return Value: "{x: 2, y: -1, direction:  
'down'}"
```

#### Example 01

```
Input: "AAAA"  
Output:  
Return Value: "{x: 0, y: -4, direction:  
'up'}"
```

#### Example 02

```
Input: ""  
Output:  
Return Value: "{x: 0, y: 0, direction:  
'up'}"
```

### Example 03

Input: "RAARA"

Output:

Return Value: "{x: 2, y: 1, direction: 'down'}"

*Tip*

(In C)

In C, you can use snprintf.

