Quest07

Subject | 1 Solution | Additional Resources (0)

Quest07

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

Macro!
There are 4 steps of compilation and macro are run at the preprocessing one.

Very useful to define `values`.

Control Center

roup formation

Progress

ubmitted

est review

Finished: approved

Go To DoCode

Access:

READ WRITE

Go To Gitea

Keep Working On This Soluti

Also working on the project

Objective: after experiencing with define, you will do something `hard` is to build algorithm with one loop and one nested loop. A great way to break down the complexity of them is to split it into smaller functions.
(my_union and inter can be done with my_string_index() for example :-)

Enjoy!

Submit directory	ex00
Submit file	my_define.c

Description

We lost a part of the following code, can you make it work! :-)

```
#include <unistd.h>

#XXXX EVEN(N) XXXXXX
#XXXX SUCCESS XXXXXX

#XXXX EVEN_MSG "I have an even number of
arguments."
#XXXX ODD_MSG XXXXXXXXXXXXXX

typedef enum s_bool
{
    XXXX,
    XXXX
} t_bool;

void    my_putchar(char c)
{
  write(1, &c, 1);
}

void    my_putstr(char *str)
{
```

yuldashe_m ergashev_a qurbonov_a atabekov_a

Just finished

Type	Project
Group Size	1 Participant
Review system	Test Review (Gandalf)
Difficulty	Initiation
Average duration	1 Week

Project's Metadata

Project

id: 38

name: quest07

visible: True

```
    int index;

    index = 0;
    while (str[index] != '\0') {
      my_putchar(str[index]);
        index++;
    }
}

t_bool  my_is_even(int nbr)
{
    return ((EVEN(nbr)) ? TRUE : FALSE);
}

void        my_define(int argc)
{
    if (my_is_even(argc) == TRUE) {
    my_putstr(EVEN_MSG);
    my_putchar('\n');
  }
    else {
    my_putstr(ODD_MSG);
    my_putchar('\n');
  }
}
```

Function prototype (c)

```
/*
**
** QWASAR.IO -- my_define
**
** @param {int} param_1
**
** @return {void}
**
*/

void my_define(int param_1)
{

}
```

Example 00

```
Input: 1
Output: I have an odd number of
arguments.

Return Value: nil
```

Example 01

```
Input: 2
Output: I have an even number of
arguments.

Return Value: nil
```

Example 02

```
Input: 3
Output: I have an odd number of
arguments.

Return Value: nil
```

Tip
(In C)
Google the following: define in C

Quest07	My Union
Submit directory	ex01
Submit file	my_union.c

Description

Write a function `my_union` that takes two strings and returns, without doubles, the
characters that appear in either one of the strings.

Function prototype (c)

```
/*
**
** QWASAR.IO -- my_union
**
** @param {char*} param_1
** @param {char*} param_2
**
** @return {char*}
**
*/

char* my_union(char* param_1, char*
param_2)
{

}
```

Example 00

```
Input: "zpadinton" &&
"paqefwtdjetyiytjneytjoeyjnejeyj"
Output:
Return Value: "zpadintoqefwjy"
```

Example 01

```
Input: "ddf6vewg64f" &&
"gtwthgdwthdwfteewhrtag6h4ffdhsd"
Output:
Return Value: "df6vewg4thras"
```

Example 02

```
Input: "rien" && "cette phrase ne cache
rien"
Output:
Return Value: "rienct phas"
```

Quest07	Inter
Submit directory	ex02
Submit file	inter.c

Description

Write a function that takes two strings and return, without doubles, the
characters that appear in both strings, in the order they appear in the first
one.

Function prototype (c)

```c
/*
**
** QWASAR.IO -- inter
**
** @param {char*} param_1
** @param {char*} param_2
**
** @return {char*}
**
*/

char* inter(char* param_1, char* param_2)
{

}
```

Example 00

```
Input: "padinton" &&
"paqefwtdjetyiytjneytjoeyjnejeyj"
Output:
Return Value: "padinto"
```

Example 01

```
Input: "ddf6vewg64f" &&
"gtwthgdwthdwfteewhrtag6h4ffdhsd"
Output:
Return Value: "df6ewg4"
```

Example 02

```
Input: "nothing" && "This sentence hides
nothing"
Output:
Return Value: "nothig"
```

Quest07	Rcapitalize
Submit directory	ex03
Submit file	rcapitalize.c

Description

Write a function that takes one string and, capitalize the last character
of each word in uppercase and the rest in lowercase.

A word is a section of string delimited by spaces/tabs or the start/end of the
string. If a word has a single letter, it must be capitalized.

A letter is a character in the set [a-zA-Z]

Function prototype (c)

```
/*
**
** QWASAR.IO -- rcapitalize
**
** @param {char*} param_1
**
** @return {char*}
**
*/

char* rcapitalize(char* param_1)
{

}
```

Example 00

```
Input: "a FiRSt LiTTlE TESt"
Output:
Return Value: "A firsT littlE tesT"
```

Example 01

```
Input: ""
Output:
Return Value: ""
```

Example 02

```
Input: "SecONd teST A LITtle BiT   Moar
comPLEX"
Output:
Return Value: "seconD tesT A littlE biT
moaR compleX"
```

Example 03

```
Input: "   But... This iS not THAT
COMPLEX"
Output:
Return Value: "   but... thiS iS noT thaT
compleX"
```

Quest07	**Is Anagram**
Submit directory	ex04
Submit file	is_anagram.c

Description

An anagram is a sequence of characters formed by rearranging the letters of
another sequence, such as 'cinema', formed from 'iceman'.

Given two strings as parameters, create a function able to tell whether or
not the first string is an anagram of the second.

Considerations:

> Be careful: the naive solution won't work on our big input, you have to find an optimized solution which will run in O(sa + sb) time (where sa is the length of a and sb length of b).
> Our tested string will always be a sequence of ascii characters between 32 and 126 inclusive.
> The bigger test we will do is on 2 sequences of 1.000.000 characteres each. It should run in less than 2 seconds.

Function prototype (c)

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

int is_anagram(char* param_1, char*
param_2)
{

}
```

Example 00

```
Input: "abcdef" && "fabcde"
Output:
Return Value: 1
```

Example 01

```
Input: "ad" && "bc"
Output:
Return Value: 0
```

Example 02

```
Input: "" && ""
Output:
Return Value: 1
```