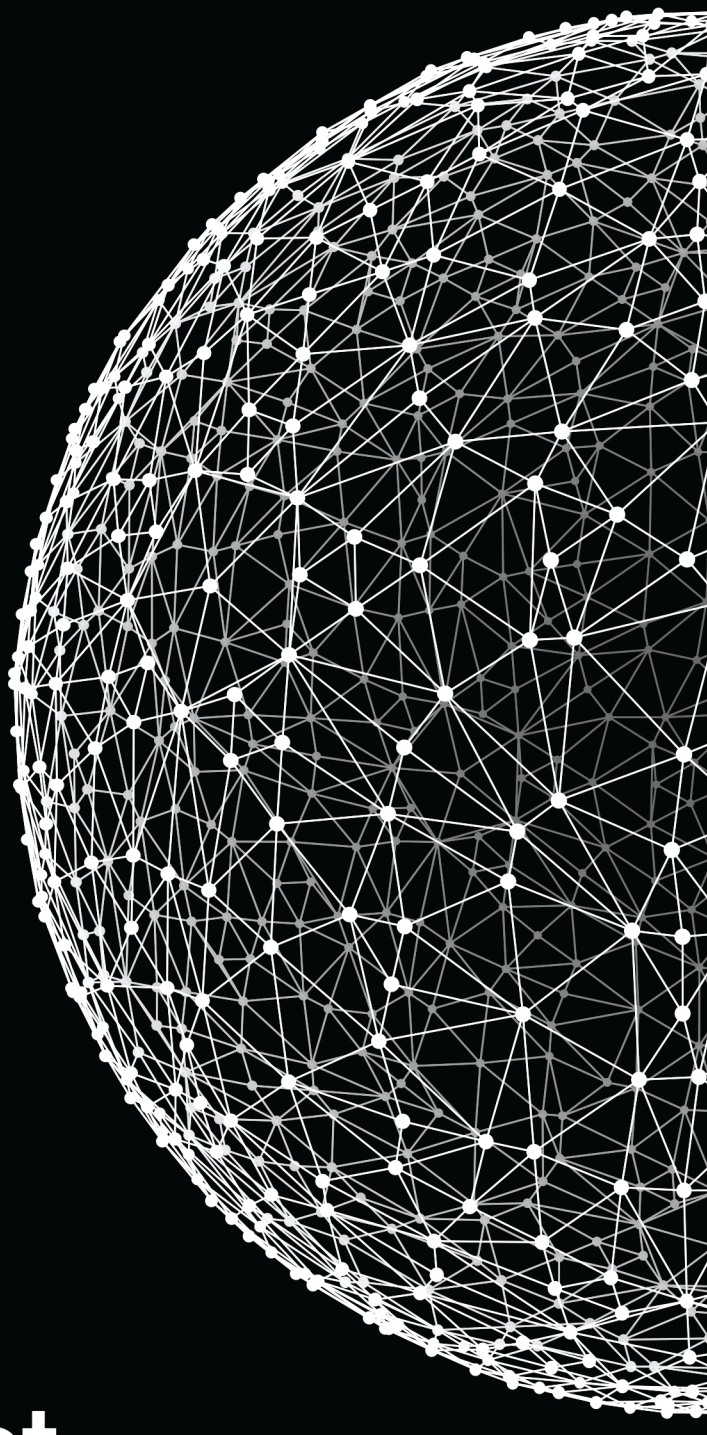
Sprint 07

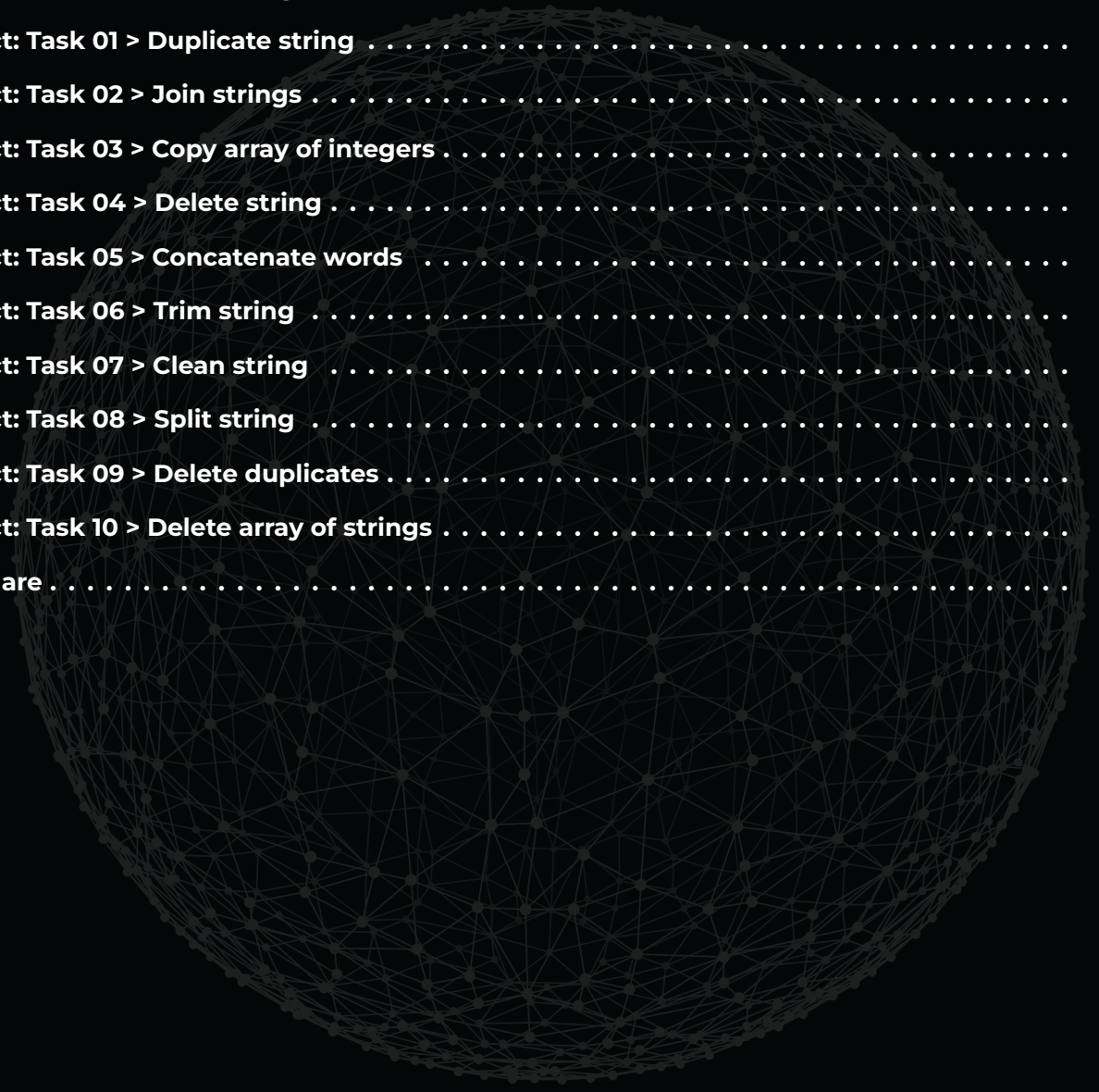
Marathon C

September 1, 2020



ucode connect

Contents



ucode connect

Engage

DESCRIPTION

Hey!

It's been a long time since you started learning C.

It's time to explore a very important topic. Namely, working with memory.
In this challenge you will learn why it is so important and powerful. Keep in mind that the success of your future education depends on the depth of knowledge and awareness of this topic.

This Sprint will expand your horizons in programming.
You will start to use the `malloc` and `free` functions to work with memory.

Let's begin.

BIG IDEA

Learning dynamical memory allocation.

ESSENTIAL QUESTION

How to allocate and work with memory in the C language?

CHALLENGE

Learn how to manage memory in the C language.

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- How does memory allocation work?
- What is `stack` and `heap`?
- What system functions can help you on with dynamical memory allocation?
- How to allocate memory for a string?
- How to free memory?
- What are `memory leaks`? And why is everyone afraid of them?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

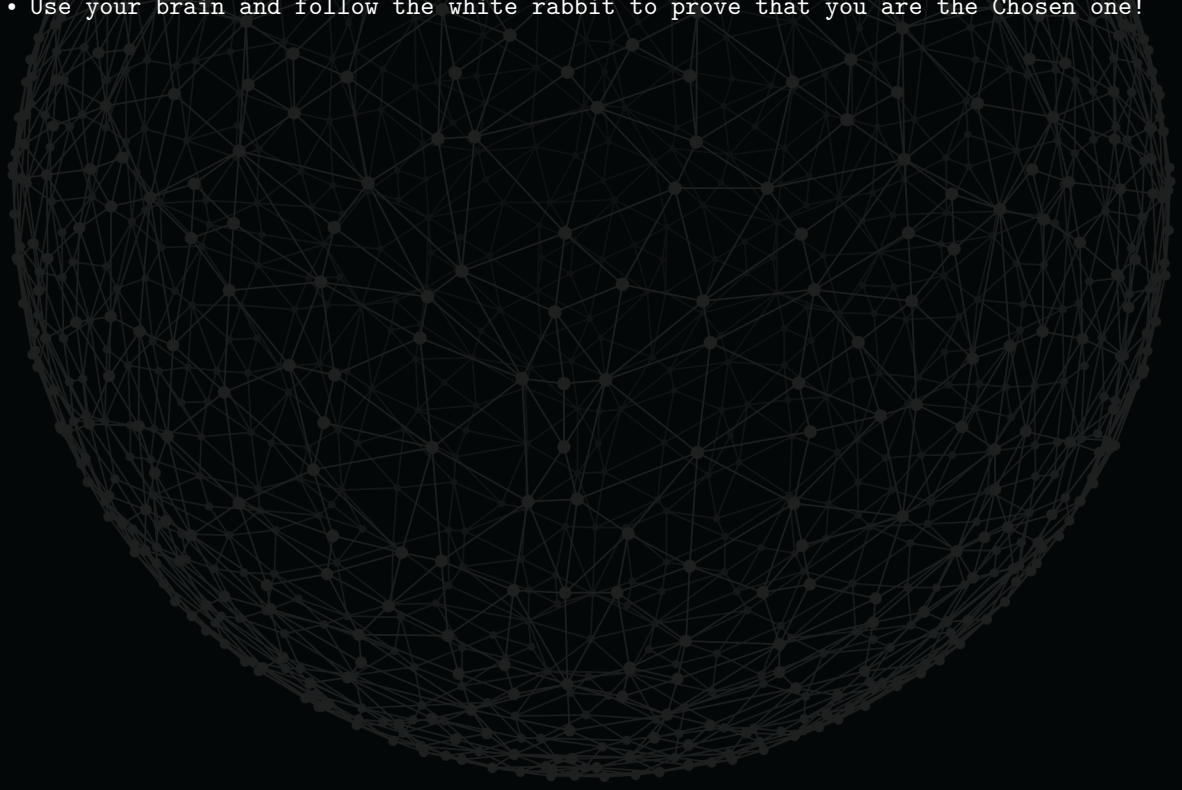
- Find information about dynamical memory allocation in C.
- Read about what `NULL` means in C and what is the difference between `NULL`, `\0` and `0`.
- Use `malloc` and `free` standard functions to experiment with memory allocation. Create a program that:
  - declares an array of characters `char *my_array;`
  - allocates 4 bytes for the array using `malloc()`
  - writes `Neo\0` to the array using `my_array = strncpy(my_array, "Neo", strlen("Neo") + 1)`
  - outputs `my_array` to the standard output
  - frees all allocated memory for `my_array` using `free()`
- Clone your git repository that is issued on the challenge page in the LMS.
- Communicate with students and share information.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- Analyze all information you have collected during the preparation stages.
- Perform only those tasks that are given in this document.

- Submit your files using the layout described in the story. Only useful files allowed, garbage shall not pass!
- Compile C-files with clang compiler and use these flags: `clang -std=c11 -Wall -Wextra -Werror -Wpedantic`.
- Your program must manage memory allocations correctly. A memory that is no longer needed must be freed, otherwise, the task is considered incomplete.
- Pay attention to what is allowed in a certain task. Use of forbidden stuff is considered a cheat and your tasks will be failed.
- Complete tasks according to the rules specified in the `Auditor`.
- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- Also, the challenge will pass automatic evaluation which is called `Oracle`.
- If you have any questions or don't understand something, ask other students or just Google it.
- Use your brain and follow the white rabbit to prove that you are the Chosen one!



Act: Task 00

NAME

New string

DIRECTORY

`t00/`

SUBMIT

`mx_strnew.c`

ALLOWED FUNCTIONS

`malloc`

DESCRIPTION

Create a function that:

- allocates memory for a string of a specific `size` and one additional byte for the terminating `'\0'`
- initializes each character with `'\0'`

RETURN

- returns the string of a specific `size` and terminated by `'\0'`
- returns `NULL` if creation fails

SYNOPSIS

```c
char *mx_strnew(const int size);
```

EXAMPLE

```c
mx_strnew(10); //returns string with size 10 and terminated by '\0'
mx_strnew(-1); //returns NULL
```

FOLLOW THE WHITE RABBIT

`man 3 malloc`

Act: Task 01

NAME

Duplicate string

DIRECTORY

`t01/`

SUBMIT

`mx_strdup.c, mx_strnew.c, mx_strlen.c, mx_strcpy.c`

ALLOWED FUNCTIONS

`malloc`

DESCRIPTION

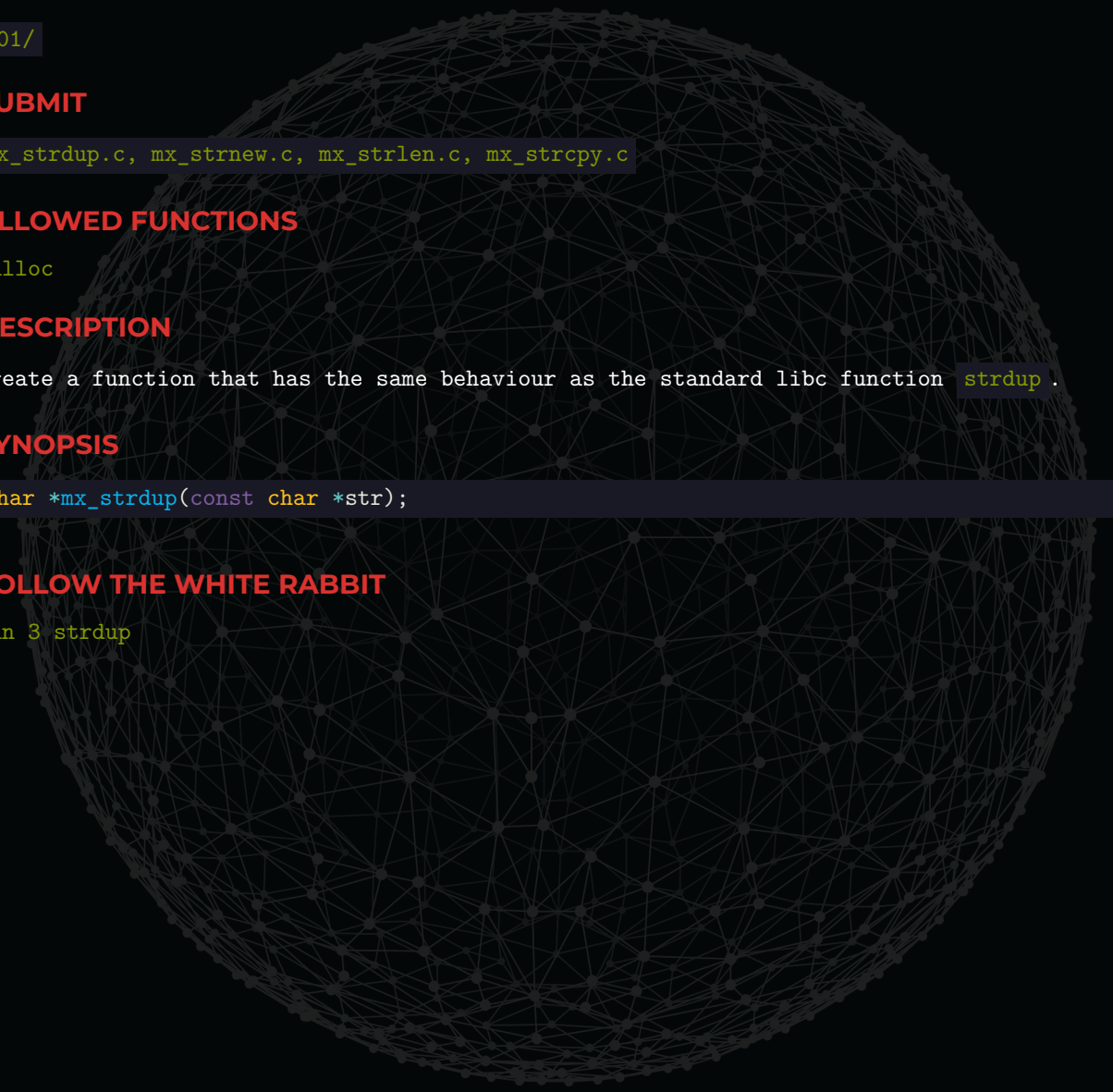
Create a function that has the same behaviour as the standard libc function `strdup`.

SYNOPSIS

```c
char *mx_strdup(const char *str);
```

FOLLOW THE WHITE RABBIT

`man 3 strdup`



Act: Task 02

NAME

Join strings

DIRECTORY

`t02/`

SUBMIT

`mx_strjoin.c, mx_strnew.c, mx_strlen.c, mx_strdup.c, mx_strcpy.c, mx_strcat.c`

ALLOWED FUNCTIONS

`malloc`

DESCRIPTION

Create a function that:

- concatenates strings `s1` and `s2` into a new string
- terminates the new string with `'\0'`

RETURN

- returns the string as a result of concatenation `s1` and `s2`
- returns the new copy of `non-NULL` parameter if one and only one of the parameters is `NULL`
- returns `NULL` if the concatenation fails

SYNOPSIS

```c
char *mx_strjoin(char const *s1, char const *s2);
```

EXAMPLE

```c
str1 = "this";
str2 = "dodge ";
str3 = NULL;
mx_strjoin(str2, str1); //returns "dodge this"
mx_strjoin(str1, str3); //returns "this"
mx_strjoin(str3, str3); //returns NULL
```

Act: Task 03

NAME

Copy array of integers

DIRECTORY

`t03/`

SUBMIT

`mx_copy_int_arr.c`

ALLOWED FUNCTIONS

`malloc`

DESCRIPTION

Create a function that copies an array of integers to a new array.

RETURN

- returns the `pointer` to the first element
- returns `NULL` if the array `src` does not exist or copying fails

SYNOPSIS

```
int *mx_copy_int_arr(const int *src, int size);
```

EXAMPLE

```
arr1 = {1, 2, 3};
arr2 = NULL;
mx_copy_int_arr(arr1, 3); //returns array [1, 2, 3]
mx_copy_int_arr(arr2, 3); //returns NULL
```

Act: Task 04

NAME

Delete string

DIRECTORY

`t04/`

SUBMIT

`mx_strdel.c`

ALLOWED FUNCTIONS

`free`

DESCRIPTION

Create a function that:

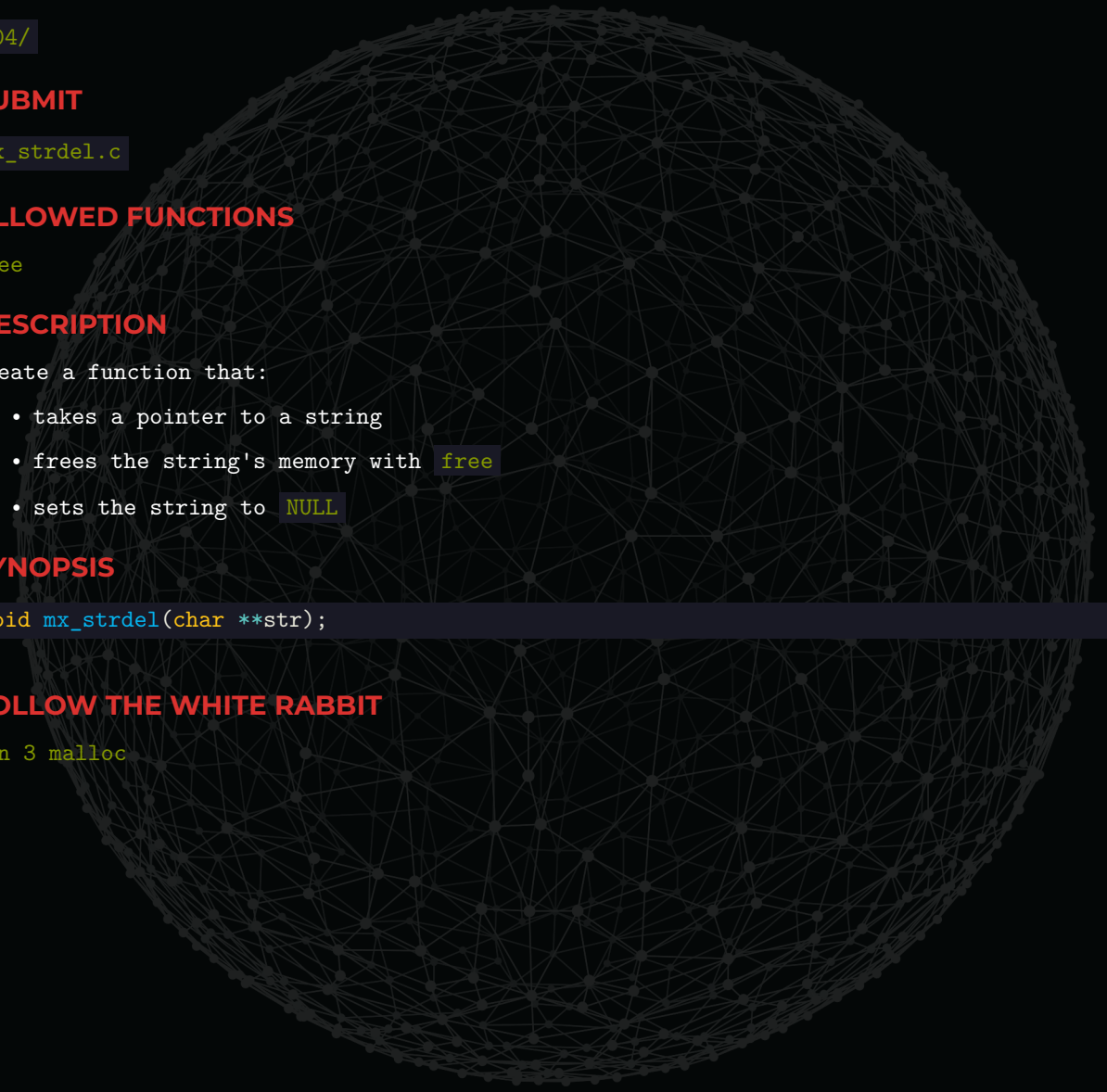
- takes a pointer to a string
- frees the string's memory with `free`
- sets the string to `NULL`

SYNOPSIS

```
void mx_strdel(char **str);
```

FOLLOW THE WHITE RABBIT

`man 3 malloc`



Act: Task 05

NAME

Concatenate words

DIRECTORY

`t05/`

SUBMIT

`mx_concat_words.c, mx_strdel.c, mx_strjoin.c, mx_strnew.c, mx_strlen.c, mx_strdup.c, mx_strcpy.c, mx_strcat.c`

ALLOWED FUNCTIONS

`malloc, free`

DESCRIPTION

Create a function that:

- concatenates the `NULL`-terminated array of words into a sentence where words are separated by a single space character
- frees all unused memory

RETURN

- returns the result of concatenation of the `NULL`-terminated array into a string
- returns `NULL` if the array of strings `words` does not exist or concatenation fails

SYNOPSIS

```c
char *mx_concat_words(char **words);
```

EXAMPLE

```c
words = {"Free", "your", "mind.", NULL};
mx_concat_words(words); //returns "Free your mind."
mx_concat_words(NULL); //returns NULL
```

SEE ALSO

Memory leaks

Act: Task 06

NAME

Trim string

DIRECTORY

`t06/`

SUBMIT

`mx_strtrim.c, mx_strdel.c, mx_isspace.c, mx_strnew.c, mx_strlen.c, mx_strncpy.c`

ALLOWED FUNCTIONS

`malloc, free`

DESCRIPTION

Create a function that:

- creates a new string without whitespace characters at the beginning and the end of the string
- frees all unused memory

RETURN

- returns a new trimmed string
- returns `NULL` if the string `str` does not exist or string trim fails

SYNOPSIS

```
char *mx_strtrim(const char *str);
```

EXAMPLE

```
name = "\f  My name... is Neo  \t\n ";
mx_strtrim(name); //returns "My name... is Neo"
```

Act: Task 07

NAME

Clean string

DIRECTORY

`t07/`

SUBMIT

`mx_del_extra_whitespaces.c, mx_strtrim.c, mx_isspace.c, mx_strncpy.c, mx_strnew.c, mx_strdel.c, mx_strlen.c`

ALLOWED FUNCTIONS

`malloc, free`

DESCRIPTION

Create a function that:

- creates a new string without whitespace characters in the beginning and/or at the end of the string
- separates words in the new string with exactly one space character
- frees all unused memory

A word is a sequence of characters separated by whitespaces.

RETURN

- returns a new created string
- returns `NULL` if the string `str` does not exist or string creation fails

SYNOPSIS

```c
char *mx_del_extra_whitespaces(const char *str);
```

EXAMPLE

```c
name = "\f  My name...      is  \r Neo  \t\n ";
mx_del_extra_whitespaces(name); //returns "My name... is Neo"
```

Act: Task 08

NAME

Split string

DIRECTORY

`t08/`

SUBMIT

`mx_strsplit.c, mx_strnew.c, mx_strncpy.c, mx_strdel.c, mx_count_words.c`

ALLOWED FUNCTIONS

`malloc, free`

DESCRIPTION

Create a function that:

- converts a string `s` to the `NULL`-terminated array of words
- frees all unused memory

A word is a sequence of characters separated by the character `c` as a delimiter.

RETURN

- returns the `NULL`-terminated array of strings
- returns `NULL` if the string `s` does not exist or conversion fails

SYNOPSIS

```c
char **mx_strsplit(char const *s, char c);
```

EXAMPLE

```c
s = "**Good bye,**Mr.*Anderson.****";
arr = mx_strsplit(s, '*');  // arr = ["Good bye,", "Mr.", "Anderson."]
s = "     Knock, knock,    Neo.    ";
arr = mx_strsplit(s, ' ');  // arr = ["Knock,", "knock,", "Neo."]
```

Act: Task 09

NAME

Delete duplicates

DIRECTORY

`t09/`

SUBMIT

`mx_del_dup_arr.c, mx_copy_int_arr.c`

ALLOWED FUNCTIONS

`malloc`

DESCRIPTION

Create a function that:

- takes an array of integers `src`, its size `src_size` and the pointer to the size of the new array `dst_size`
- initializes `dst_size` by the size of the array without duplicates
- creates the new array without duplicates

RETURN

- returns a new array without duplicates
- returns `NULL` if the array `src` does not exist or creation fails

SYNOPSIS

```
int *mx_del_dup_arr(int *src, int src_size, int *dst_size);
```

EXAMPLE

```
//initial array
{1, 2, 2, 8, 4, 6, 8, 9, -4, 3, 4}
//new array without duplicates
{1, 2, 8, 4, 6, 9, -4, 3}
```

Act: Task 10

NAME

Delete array of strings

DIRECTORY

`t10/`

SUBMIT

`mx_del_strarr.c, mx_strdel.c`

ALLOWED FUNCTIONS

`free`

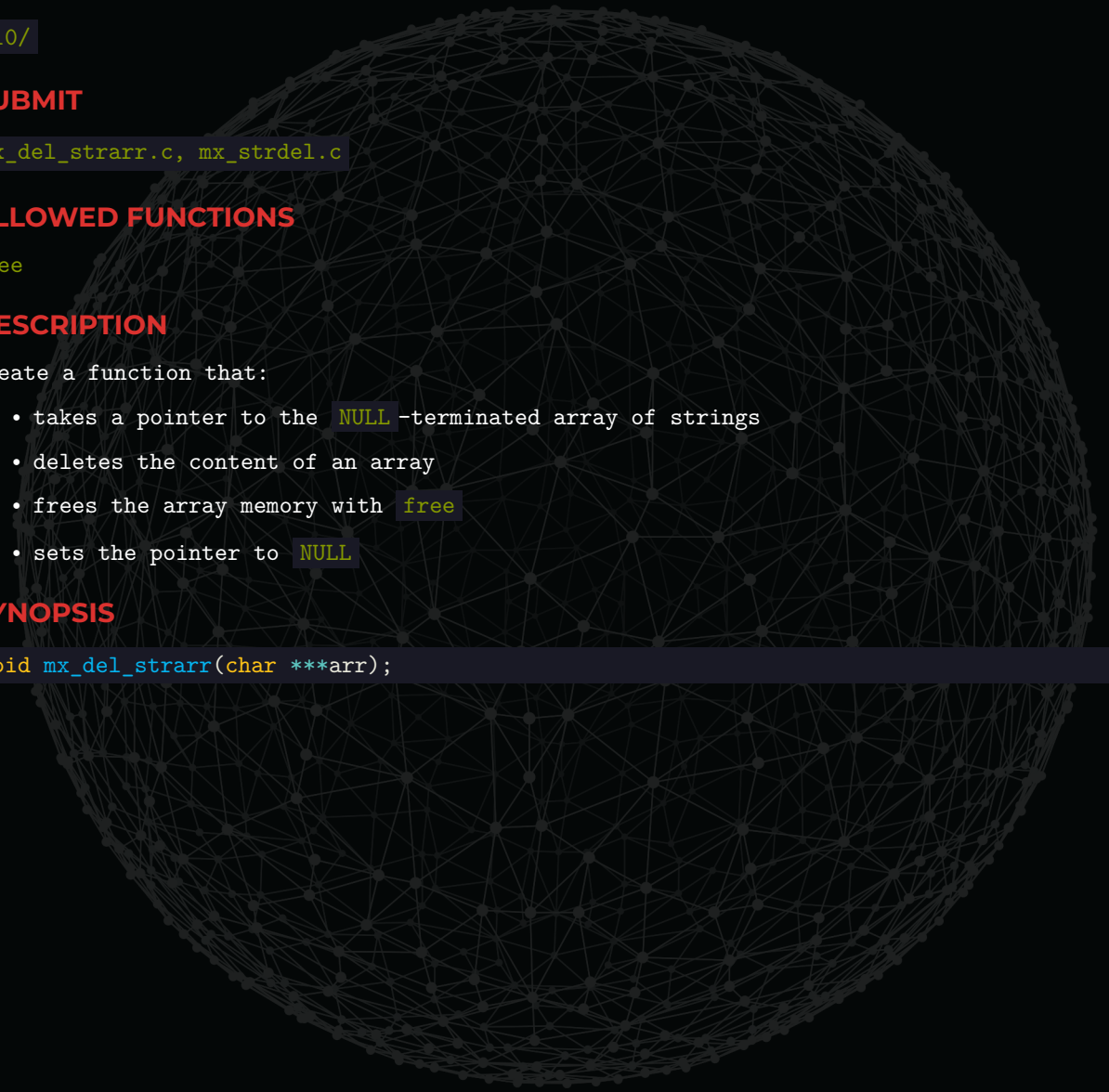
DESCRIPTION

Create a function that:

- takes a pointer to the `NULL`-terminated array of strings
- deletes the content of an array
- frees the array memory with `free`
- sets the pointer to `NULL`

SYNOPSIS

```c
void mx_del_strarr(char ***arr);
```



Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Telegraph - create a post that you can easily share on Telegram
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.