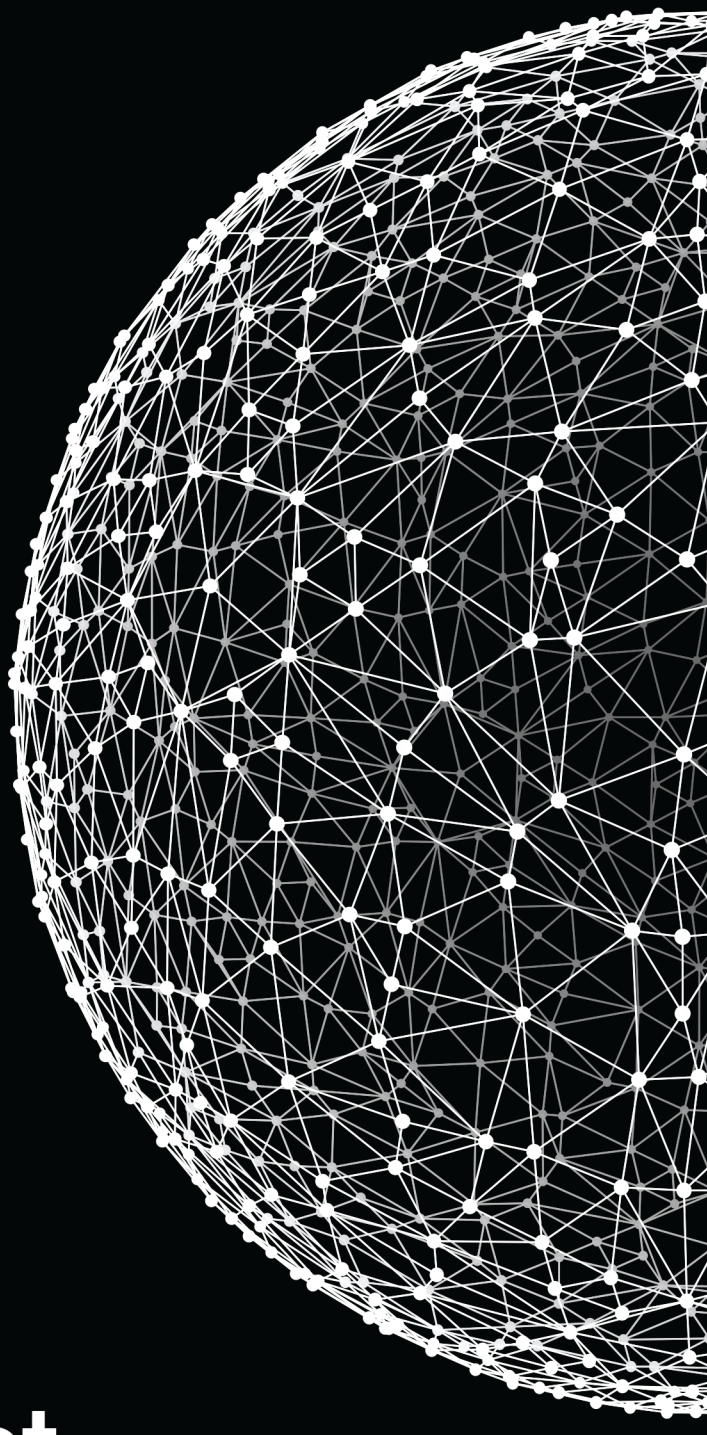
Sprint 06

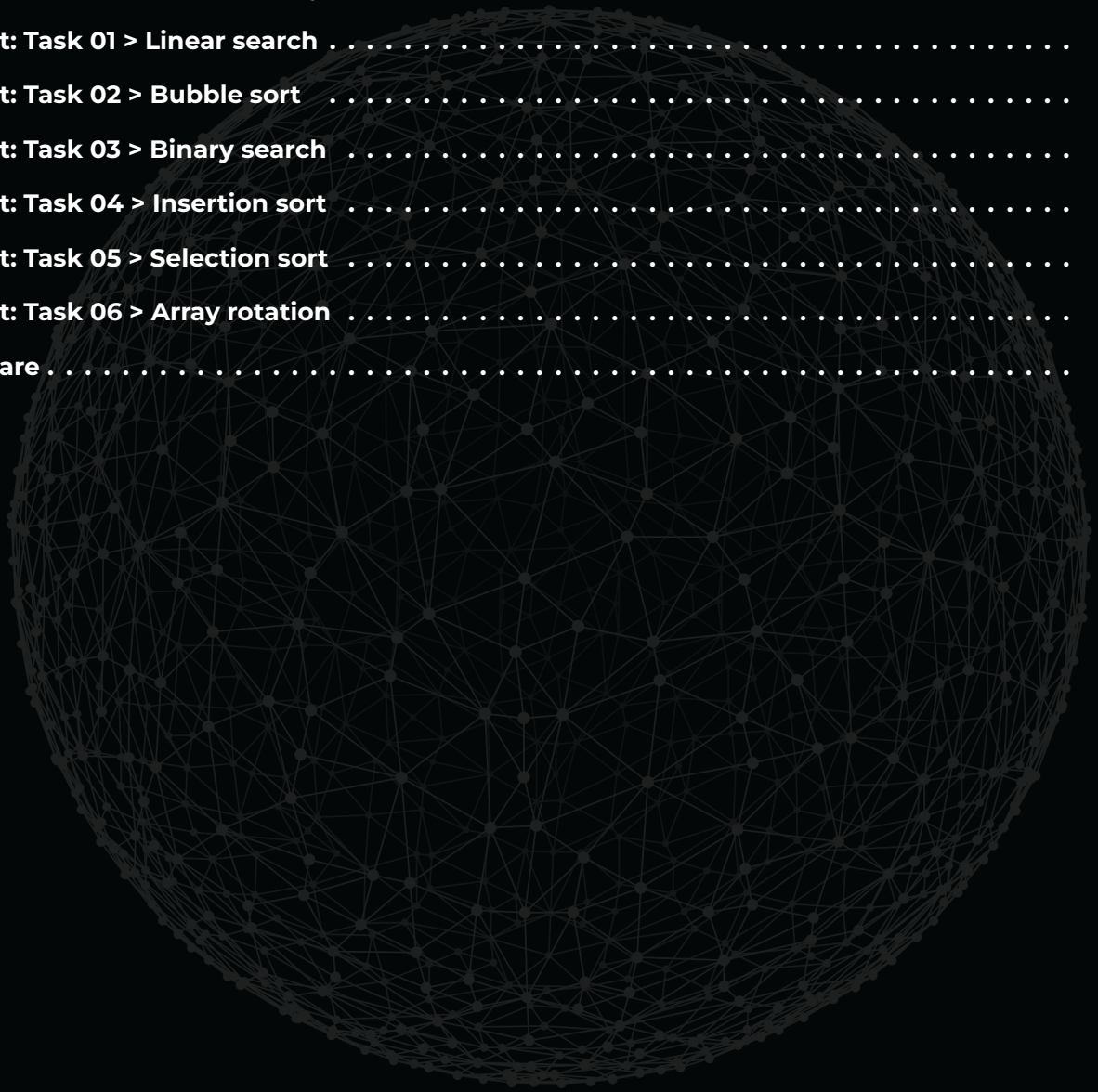
Marathon C

September 1, 2020



ucode connect

Contents



ucode connect

Engage

DESCRIPTION

Hi!

New Sprint!
Ready, steady, go!

The tasks of this Sprint are aimed at studying sorting algorithms and optimization functions in the form of a library. Some of these algorithms you've already used before, so you have the opportunity to consolidate existing knowledge and gain more.

Sort them all!

BIG IDEA

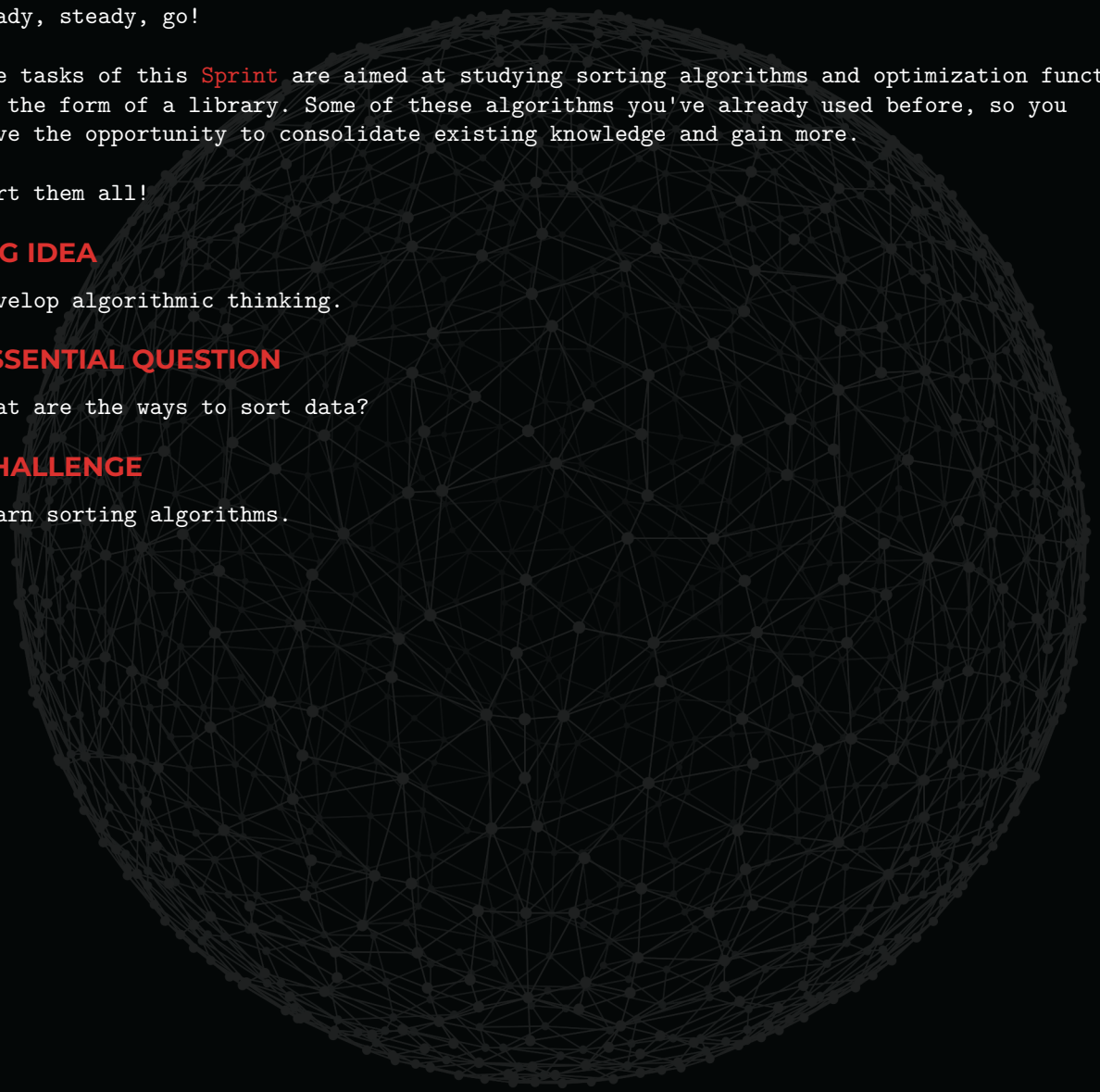
Develop algorithmic thinking.

ESSENTIAL QUESTION

What are the ways to sort data?

CHALLENGE

Learn sorting algorithms.



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 was the previous Sprint? Is everything clear?
- What is an algorithm?
- What is sorting?
- What sorting algorithms do you know?
- How useful is each sorting algorithm? For what can each of them be used?
- What is a library in programming languages?
- What are the benefits of creating and using libraries?
- How to create your own library? How to use it in your program?

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.

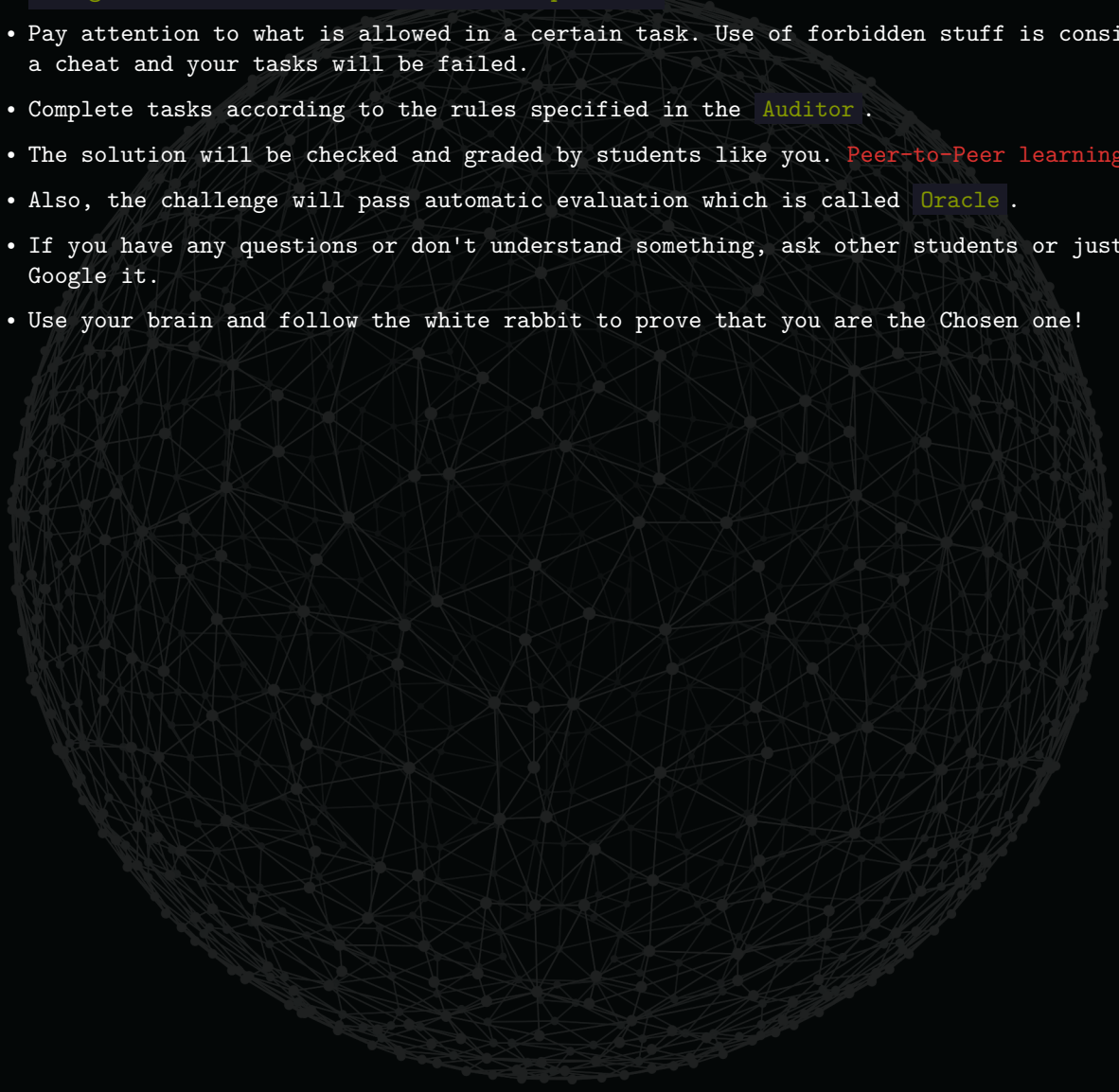
- Create a program that:
  - takes an array of integers as arguments
  - sorts the array in ascending order using bubble sort
  - prints the sorted array to the standard output
- Create your own library of several functions. For example:
  - take `mx_printchar`, `mx_printstr` and `mx_printint`
  - make object files from `.c` files of specified functions (`man clang`)
  - create a library archive from object files using `ar` (`man ar`)
- Compile your program from the first paragraph using your new library
  - use `clang your_program.c library.a -o test_program`
  - as you can see, now you can use your library to compile instead of every necessary function
- Clone your git repository that is issued on the challenge page in the LMS.
- If you managed all that, you are ready for the `task00`. Let's go!

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.

- 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` .
- 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

Matrix library

DIRECTORY

`t00/`

SUBMIT

`create_minilibmx.sh, mx_printchar.c, mx_printint.c, mx_printstr.c, mx_strcpy.c, mx_strlen.c, mx_strcmp.c, mx_isdigit.c, mx_isspace.c, mx_atoi.c, other useful functions`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a shell script that creates a library called `minilibmx.a` with functions listed below:

- `mx_printchar`
- `mx_printint`
- `mx_printstr`
- `mx_strcpy`
- `mx_strlen`
- `mx_strcmp`
- `mx_isdigit`
- `mx_isspace`
- `mx_atoi`

Of course, you can add some other useful functions into your library.

SYNOPSIS

```c
void mx_printchar(char c);
void mx_printint(int n);
void mx_printstr(const char *s);
char *mx_strcpy(char *dst, const char *src);
int mx_strlen(const char *s);
int mx_strcmp(const char *s1, const char *s2);
bool mx_isdigit(char c);
bool mx_isspace(char c);
int mx_atoi(const char *str);
```

Act: Task 01

NAME

Linear search

DIRECTORY

`t01/`

SUBMIT

`mx_linear_search.c, mx_strcmp.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that:

- searches the string `s` in the sorted NULL-terminated array `arr`
- uses the `linear search` algorithm

RETURN

- returns the `index` of the found string in the array
- returns `-1` in case of errors or if the string has not been found

SYNOPSIS

```c
int mx_linear_search(char **arr, const char *s);
```

EXAMPLE

```c
arr = {"222", "Abcd", "aBc", "ab", "az", "z", NULL};
mx_linear_search(arr, "z");  //returns 5
mx_linear_search(arr, "aBc");  //returns 2
```

SEE ALSO

Linear search

Act: Task 02

NAME

Bubble sort

DIRECTORY

`t02/`

SUBMIT

`mx_bubble_sort.c, mx_strcmp.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that:

- sorts an array of strings in place in lexicographical order
- uses the `bubble sort` algorithm

RETURN

Returns the number of swap operations.

SYNOPSIS

```
int mx_bubble_sort(char **arr, int size);
```

EXAMPLE

```
arr = {"abc", "xyz", "ghi", "def"};
mx_bubble_sort(arr, 4);  //returns 3

arr = {"abc", "acb", "a"};
mx_bubble_sort(arr, 3);  //returns 2
```

SEE ALSO

Bubble sort

Act: Task 03

NAME

Binary search

DIRECTORY

`t03/`

SUBMIT

`mx_binary_search.c, mx_strcmp.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that:

- searches the string `s` in the array `arr` with the given `size`
- uses the binary search algorithm assuming that the input array has already been sorted in lexicographical order

RETURN

- returns the `index` of the found string in the array
- returns `-1` in case of errors or if the string has not been found
- assigns the number of required iterations to the `count` pointer

SYNOPSIS

```c
int mx_binary_search(char **arr, int size, const char *s, int *count);
```

EXAMPLE

```c
arr = {"222", "Abcd", "aBc", "ab", "az", "z"};
count = 0;
mx_binary_search(arr, 6, "ab", &count);  //returns 3 and count = 3
count = 0;
mx_binary_search(arr, 6, "aBc", &count);  //returns 2 and count = 1
count = 0;
mx_binary_search(arr, 6, "aBz", &count);  //returns -1 and count = 0
```

SEE ALSO

Binary search

Act: Task 04

NAME

Insertion sort

DIRECTORY

`t04/`

SUBMIT

`mx_insertion_sort.c, mx_strlen.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that:

- sorts an array of strings in place by length in ascending order
- uses the `insertion sort` algorithm

RETURN

Returns the number of shift operations.

SYNOPSIS

```c
int mx_insertion_sort(char **arr, int size);
```

EXAMPLE

```c
arr = {"12aaaaaaaaaa", "11aaaaaaaaa", "13aaaaaaaaaaa", "5aaaa", "6aaaaa"};
mx_insertion_sort(arr, 5);  //returns 7

arr = {"abc", "ab", "aaaaa", "aaaa", "aaa"};
mx_insertion_sort(arr, 5);  //returns 4
```

SEE ALSO

Insertion sort

Act: Task 05

NAME

Selection sort

DIRECTORY

`t05/`

SUBMIT

`mx_selection_sort.c, mx_strcmp.c, mx_strlen.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that:

- sorts an array of strings in place by length in ascending order
- uses the `selection sort` algorithm
- sorts strings in lexicographical order if their length is equal

RETURN

Returns the number of swap operations.

SYNOPSIS

```c
int mx_selection_sort(char **arr, int size);
```

EXAMPLE

```c
arr = {"Abcd", "a", "aBc", "abc", "Z", "z", "AbCd"};
mx_selection_sort(arr, 7);  //returns 5

arr = {"Z", "Abcd", "a", "aBc", "z", "abc", "AbCd"};
mx_selection_sort(arr, 7);  //returns 4
```

SEE ALSO

Selection sort

Act: Task 06

NAME

Array rotation

DIRECTORY

`t06/`

SUBMIT

`mx_arr_rotate.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that:

- rotates an array `arr` of `size` by `shift` elements
- rotates the array to the left if `shift` is negative and rotates to the right in the opposite case

SYNOPSIS

```
void mx_arr_rotate(int *arr, int size, int shift);
```

EXAMPLE

```
arr = {1, 2, 3, 4, 5};
mx_arr_rotate(arr, 5, 2); // arr = {4, 5, 1, 2, 3}

arr = {1, 2, 3, 4, 5};
mx_arr_rotate(arr, 5, -2); // arr = {3, 4, 5, 1, 2}

arr = {1, 2, 3, 4, 5};
mx_arr_rotate(arr, 5, 11);  // arr = {5, 1, 2, 3, 4}
```

SEE ALSO

Array rotation

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.