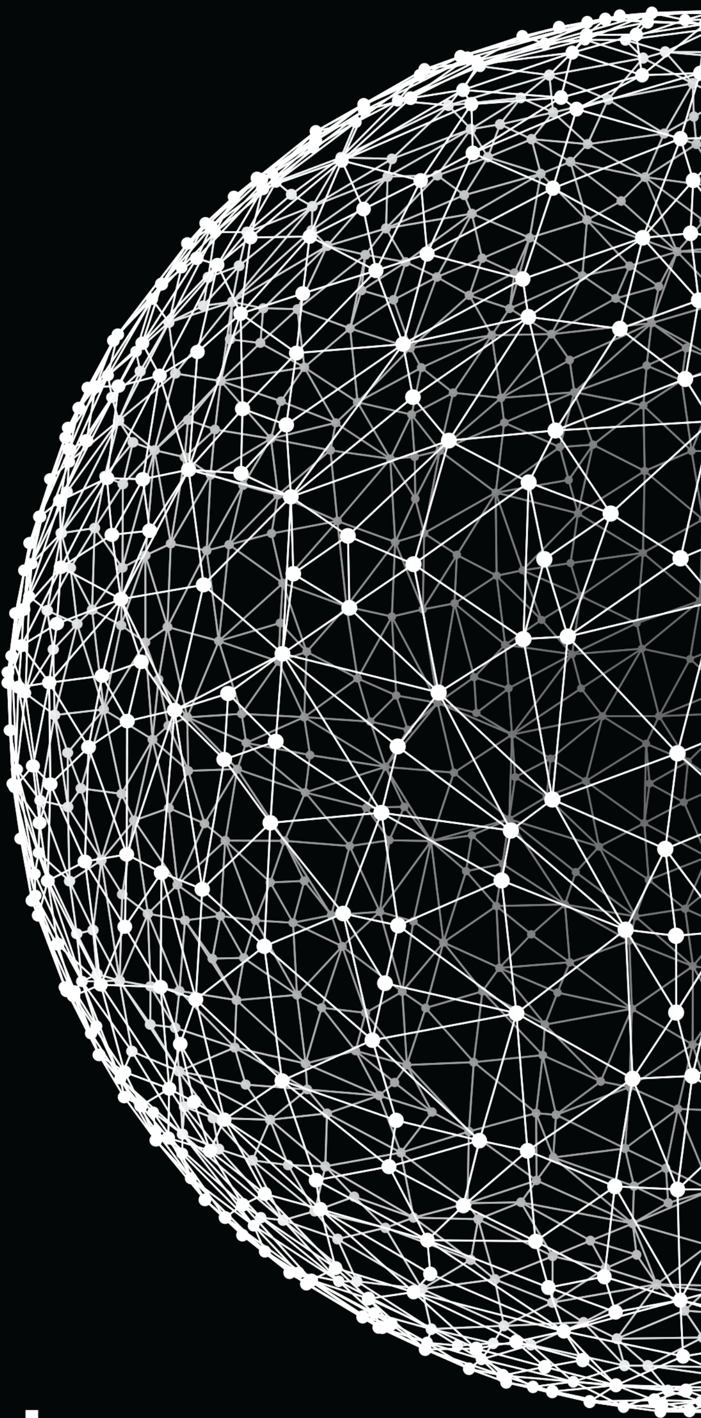
Checkpoint 00

Marathon C

September 15, 2021



ucode connect

Contents

ucode connect

Challenge Based Learning

- The Checkpoint is a reflection of what you've learned during your time at ucode connect so far. It is a tool for you to evaluate your understanding of the material, and for us to make sure all students are on the right track.
- You will help yourself by completing this task honestly. This challenge will help you understand whether you are moving in the right direction. You have an opportunity to identify areas that you know well, and those that you need to spend more time on. Also, try to reflect on whether you have used the possibilities of P2P learning to the max, and think about how to improve in that area.
- During Checkpoint you can not speak with other students, you do not have a right to chat, listen to music, make noise, or generally do anything that may disturb the other students in any way.
- Your phones and other technological devices must be turned off and put away. If a phone rings, you will be disqualified from the Checkpoint and kicked out immediately.
- Do not use the internet. Complete the Checkpoint on your own to get the best result.
- To achieve the best result for yourself, you must have only a piece of paper and a pen.
- Only in the case of a technical issue, problem understanding the story, or other questions related to the Checkpoint, you can write to the channel #___ucode_connect_team (but don't ask how to solve a task, that's on you). The ucode connect team will quickly respond to your request.
- 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!
- Read `man` to carry out some tasks if you need it.
- Compile C-files with clang compiler and use these flags: `-std=c11 -Wall -Wextra -Werror -Wpedantic`.
- Pay attention to what is allowed in a certain task. You may recode any other function you think is necessary. Use of forbidden stuff is considered a cheat and your tasks will be failed.
- The evaluation is fully automated. Tasks will pass automatic evaluation which is called `Oracle`.
- An important part of your study is an analysis of mistakes to avoid them in the future. To help you to revise your mistakes, the read-only access to your Checkpoint repository will stay open even after the Checkpoint is finished. You can find the link to the repository at the challenge page in the LMS.
  Use this opportunity wisely. Try completing the tasks that you've failed during the Checkpoint.

Act Basic: Task 00

NAME

Only printable

DIRECTORY

`t00/`

SUBMIT

`mx_only_printable.c`

ALLOWED FUNCTIONS

`write`

DESCRIPTION

Create a function that outputs all printable characters in reverse order to the standard output followed by a newline.

Hint: Space is a printable character.

SYNOPSIS

```
void mx_only_printable(void);
```

FOLLOW THE WHITE RABBIT

`man ascii`

Act Basic: Task 01

NAME

Print string

DIRECTORY

`t01/`

SUBMIT

`mx_printstr.c`

ALLOWED FUNCTIONS

write

DESCRIPTION

Create a function that outputs a string of characters to the standard output.

SYNOPSIS

```
void mx_printstr(const char *s);
```

FOLLOW THE WHITE RABBIT

man 2 write

Act Basic: Task 02

NAME

Is white-space?

DIRECTORY

`t02/`

SUBMIT

`mx_isspace.c`

ALLOWED FUNCTION

None

DESCRIPTION

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

SYNOPSIS

```
bool mx_isspace(char c);
```

EXAMPLE

```
mx_isspace(' ');  //returns 1
```

FOLLOW THE WHITE RABBIT

`man isspace`

Act Basic: Task 03

NAME

Print integer

DIRECTORY

`t03/`

SUBMIT

`mx_printint.c`

ALLOWED FUNCTIONS

`write`

DESCRIPTION

Create a function that outputs integer values to the standard output.

SYNOPSIS

```
void mx_printint(int n);
```

EXAMPLE

```
mx_printint(25); //prints 25
mx_printint(2147483647); //prints 2147483647
```

ucode connect

Act Basic: Task 04

NAME

Reverse string

DIRECTORY

`t04/`

SUBMIT

`mx_str_reverse.c`

ALLOWED FUNCTION

None

DESCRIPTION

Create a function that reverses a string using pointers.

SYNOPSIS

```
void mx_str_reverse(char *s);
```

EXAMPLE

```
str = "game over";
mx_str_reverse(str); //'str' now is "revo emag"
```

Act Basic: Task 05

NAME

Compare strings

DIRECTORY

`t05/`

SUBMIT

`mx_strcmp.c`

ALLOWED FUNCTION

None

DESCRIPTION

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

SYNOPSIS

```
int mx_strcmp(const char *s1, const char *s2);
```

FOLLOW THE WHITE RABBIT

`man 3 strcmp`

ucode connect

Act Basic: Task 06

NAME

Copy string

DIRECTORY

`t06/`

SUBMIT

`mx_strcpy.c`

ALLOWED FUNCTION

None

DESCRIPTION

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

SYNOPSIS

```
char *mx_strcpy(char *dst, const char *src);
```

FOLLOW THE WHITE RABBIT

`man 3 strcpy`

Act Basic: Task 07

NAME

Sort array

DIRECTORY

`t07/`

SUBMIT

`mx_sort_arr_int.c`

ALLOWED FUNCTION

None

DESCRIPTION

Create a function that sorts an array of integers in ascending order.

SYNOPSIS

```
void mx_sort_arr_int(int *arr, int size);
```

EXAMPLE

```
arr = {3, 55, -11, 1, 0, 4, 22};
mx_sort_arr_int(arr, 7); //arr now is '{-11, 0, 1, 3, 4, 22, 55}'
```

Act Basic: Task 08

NAME

ASCII to integer

DIRECTORY

`t08/`

SUBMIT

`mx_atoi.c`

ALLOWED FUNCTION

None

DESCRIPTION

Create a function that converts an ASCII string to an integer.

SYNOPSIS

```
int mx_atoi(const char *str);
```

FOLLOW THE WHITE RABBIT

man atoi