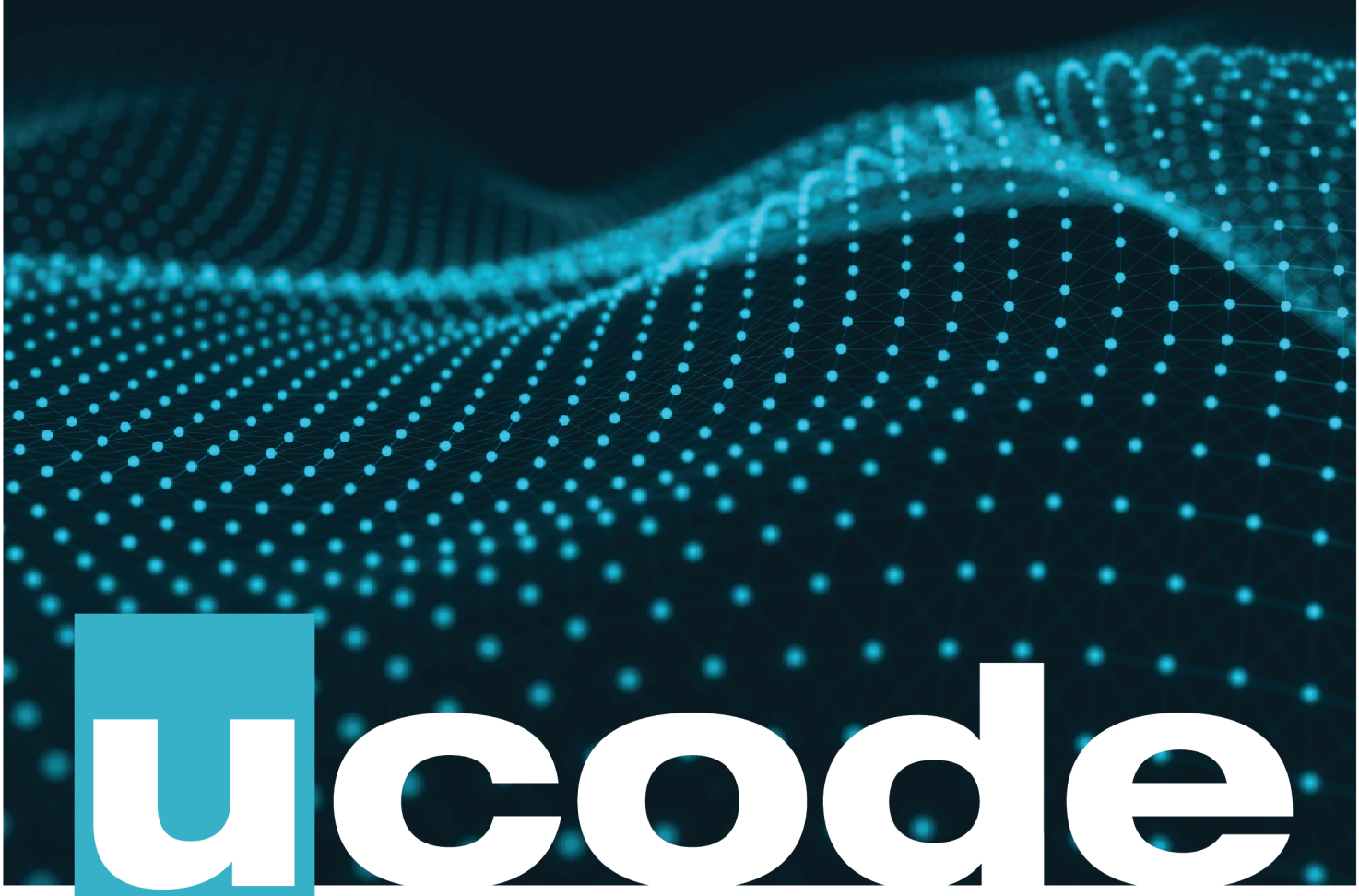
Marathon C

Sprint 08

June 7, 2019



ucode

Contents

Challenge Based Learning 2

Task 00 > Header intro 3

Task 01 > Structure intro 4

Task 02 > Decimal to hex 6

Task 03 > Hex to decimal 7

Task 04 > Get address 8

Task 05 > Neo's choice 9

Task 06 > Matrix need a new agent 11

Task 07 > More agents!!! 13

Task 08 > Ex-ter-mi-nate agents 14

Task 09 > Smiths 15

ucode

Challenge Based Learning

1. Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
2. Perform only those tasks that are given in the story.
3. You should submit only the specified files in the required directory and nothing else. In case you are allowed to submit any files to complete the task you should submit only useful files. Garbage shall not pass.
4. You should compile C files with clang compiler and use these flags: `-std=c11 -Wall -Wextra -Werror -Wpedantic`.
5. Your program must manage memory allocations correctly. Memory which is no longer needed must be released otherwise the task is considered as incomplete.
6. You should use only functions which allowed in a certain task.
7. Usage of forbidden functions is considered as cheat and your challenge will be failed.
8. You must complete tasks according to the rules specified in `the Auditor`.
9. Your exercises will be checked and graded by students. The same as you. `Peer-to-Peer (P2P) learning`.
10. Also, your exercises will pass automatic evaluation which is called `Oracle`.
11. Got a question or you do not understand something? Ask the students or just Google that.
12. Use your brain and follow the white rabbit to prove that you are the Chosen one!!!

Task 00

NAME

Header intro

DIRECTORY

`t00/`

SUBMIT

`header.h`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a header that contains prototypes of your library functions. Prototypes of the next functions are mandatory:

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

We will test our functions by using your header.

FOLLOW THE WHITE RABBIT

https://gcc.gnu.org/onlinedocs/cpp/Once-Only-Headers.html
https://www.tutorialspoint.com/cprogramming/c_header_files.htm

Task 01

NAME

Structure intro

DIRECTORY

`t01/`

SUBMIT

`duplicate.h, mx_del_dup_sarr.c, mx_copy_int_arr.c`

ALLOWED FUNCTIONS

`malloc`

DESCRIPTION

In this task you need to:

- rewrite the function `mx_del_dup_arr` using the structure;
- develop a function that creates new structure with new array without duplicates and its size;
- create a header which includes all the necessary headers, prototypes and structure `s_intarr`.

RETURN VALUES

- new structure with new array without duplicates and its size;
- `NULL` if structure does not exist or creation fails.

SYNOPSIS

```
typedef struct s_intarr
{
    int *arr;
    int size;
} t_intarr;
```

```
t_intarr *mx_del_dup_sarr(t_intarr *src);
```

FOLLOW THE WHITE RABBIT

https://www.programiz.com/c-programming/c-structures
http://lmgtfy.com/?q=c+programming+structure

Task 02

NAME

Decimal to hex

DIRECTORY

`t02/`

SUBMIT

`mx_nbr_to_hex.c, mx_strnew.c`

ALLOWED FUNCTIONS

`malloc`

DESCRIPTION

Create a function that converts an `unsigned long` number into a hexadecimal.

RETURN VALUES

Returns number converted into hexadecimal string.

SYNOPSIS

```
char *mx_nbr_to_hex(unsigned long nbr);
```

EXAMPLE

```
mx_nbr_to_hex(52); //returns "34"
mx_nbr_to_hex(1000); //returns "3e8"
```

Task 03

NAME

Hex to decimal

DIRECTORY

`t03/`

SUBMIT

`mx_hex_to_nbr.c, mx_isdigit.c, mx_isalpha.c, mx_islower.c, mx_isupper.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that converts a hexadecimal string into an `unsigned long` number.

RETURN VALUES

Return `unsigned long` number.

SYNOPSIS

```
unsigned long mx_hex_to_nbr(const char *hex);
```

EXAMPLE

```
hex_to_nbr("C4"); //returns 196
hex_to_nbr("FADE"); //returns 64222
hex_to_nbr("fffffffffffff"); //returns 281474976710655
```

Task 04

NAME

Get address

DIRECTORY

`t04/`

SUBMIT

`mx_get_address.c, mx_nbr_to_hex.c, mx_strcpy.c, mx_strlen.c, mx_strnew.c`

ALLOWED FUNCTIONS

`malloc, free`

DESCRIPTION

Create a function that takes pointer and returns its memory address in a hexadecimal format with prefix `"0x"`.

RETURN VALUES

Address of the pointer as a string

SYNOPSIS

```
char *mx_get_address(void *p);
```

Task 05

NAME

Neo's choice

DIRECTORY

`t05/`

SUBMIT

`choice.h`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a header `choice.h` with which the program compiles and works.
If Neo chooses:

- red pill program prints `"Follow me!"`;
- blue `"Perhaps I was wrong about you, Neo."`;
- something other `"Are you sure about that?"`.

SYNOPSIS

```c
#include "choice.h"

t_phrase *choice(int pill) {
    char *res;

    if (pill == MX_RED_PILL) {
        res = mx_strdup(MX_SUCCESS_PHRASE);
    }
    else if (pill == MX_BLUE_PILL) {
        res = mx_strdup(MX_FAIL_PHRASE);
    }
    else {
        res = mx_strdup(MX_UNDEFINED_PHRASE);
    }
        return res;
}

int main(void) {
    t_phrase *phrase1 = choice(MX_RED_PILL);
    t_phrase *phrase2 = choice(MX_BLUE_PILL);
    t_phrase *phrase3 = choice((MX_RED_PILL + MX_BLUE_PILL) * 2);

    printf("%s\n", phrase1);
    printf("%s\n", phrase2);
    printf("%s\n", phrase3);
    return 0;
}
```

Task 06

NAME

Matrix need a new agent

DIRECTORY

`t06/`

SUBMIT

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

ALLOWED FUNCTIONS

`malloc`

DESCRIPTION

1. The Matrix chose you to develop new agent creator.
2. The Matrix will use your function with its own header `agent.h` with structure `s_agent`.
3. Function must allocate new memory (duplicate) for `name` parameter.

RETURN VALUES

- `pointer` to allocated memory of new structure;
- `NULL` if `name` is `NULL` or agent creation fails function.

SYNOPSIS

```
typedef struct s_agent
{
    char *name;
    int power;
    int strength;
} t_agent;
```

```
t_agent *mx_create_agent(char *name, int power, int strength);
```

EXAMPLE

```
agent = mx_create_agent("Smith", 150, 66);
//agent->name is "Smith"
//agent->power is 150
//agent->strength is 66
```

Task 07

NAME

More agents!!!

DIRECTORY

`t07/`

SUBMIT

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

ALLOWED FUNCTIONS

`malloc, free`

DESCRIPTION

1. Write a function that creates a `NULL-terminated` array of pointers to agents.
2. Data for each agent are stored in 3 different arrays `name`, `power` and `strength`.
3. Each agent characteristic is placed at the same index in a respective array.
4. Of course, you need to use structure from `agent.h`.

RETURN VALUES

- `NULL-terminated` array of agents;
- `NULL` if one of the parameters of function is `NULL` or agent creation fails.

SYNOPSIS

```
t_agent **mx_create_new_agents(char **name, int *power, int *strength, int count);
```

EXAMPLE

```
names = {"Thompson", "Smith", "Colson"};
powers = {33, 66, 99};
strengths = {133, 166, 196};
mx_create_new_agents(names, powers, strengths, 3); //returns 't_agent' type array
```

Task 08

NAME

Ex-ter-mi-nate agents

DIRECTORY

`t08/`

SUBMIT

`mx_exterminate_agents.c`

ALLOWED FUNCTIONS

`free`

DESCRIPTION

Create a function that:

- frees a `NULL-terminated` array of agents;
- frees contents of each agent;
- sets pointer to `NULL`;
- uses structure from `agent.h`.

SYNOPSIS

```
void mx_exterminate_agents(t_agent ***agents);
```

Task 09

NAME

Smiths

DIRECTORY

`t09/`

SUBMIT

`mx_only_smiths.c, mx_strcmp.c, mx_exterminate_agents.c, mx_create_agent.c, mx_strdup.c, mx_strnew.c, mx_strlen.c, mx_strcpy.c`

ALLOWED FUNCTIONS

`malloc, free`

DESCRIPTION

1. Write a function that creates a new `NULL-terminated` array of pointers to agents.
2. New array has only agents with the name `Smith` and strength lower than `strength` parameter of function.
3. Input `agents` must be exterminated.
4. Of course, you need to use structure from `agent.h`.

RETURN VALUES

- new filtered array;
- `NULL` if original array is `NULL` or creation of new array fails.

SYNOPSIS

```
t_agent **mx_only_smiths(t_agent **agents, int strength);
```

EXAMPLE

```
agents[0] = mx_create_agent("Smith", 150, 166);
agents[1] = mx_create_agent("Brown", 147, 57);
agents[2] = mx_create_agent("Smith", 151, 65);
agents[3] = mx_create_agent("Smith", 123, 321);
agents[4] = NULL;
mx_only_smiths(agents, 100); //returns array with 1 element
```