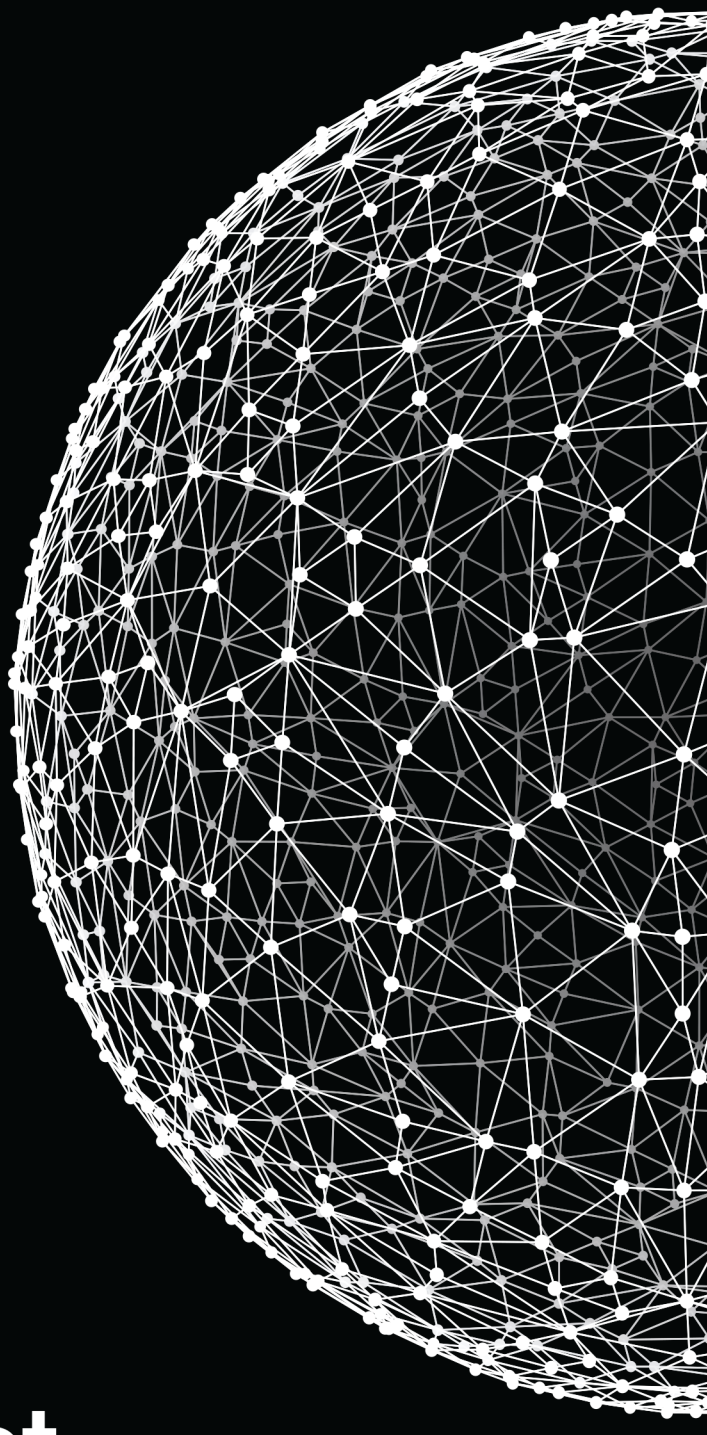
Race01: Part of the Matrix

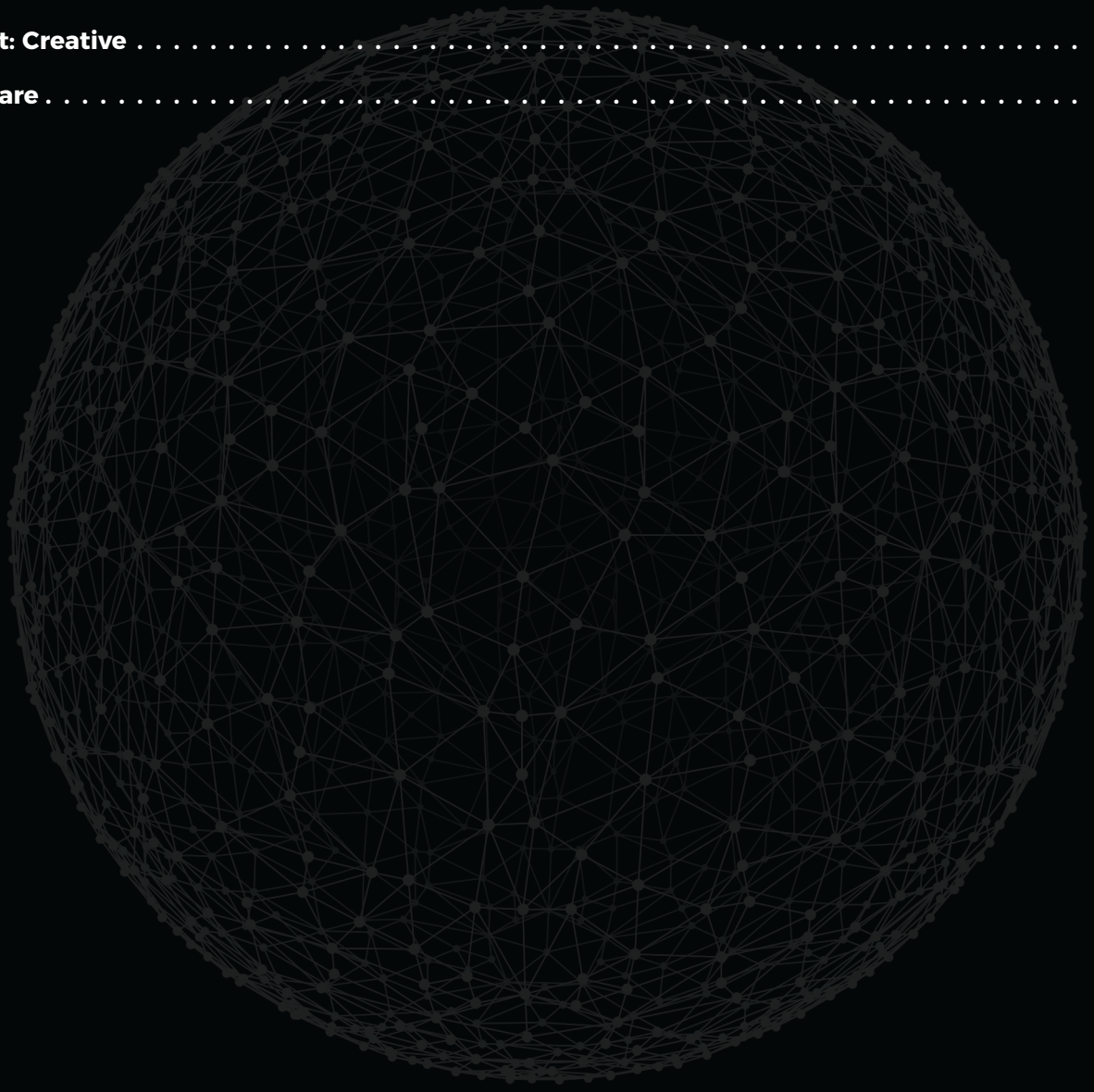
Marathon C

September 3, 2020



ucode connect

Contents



ucode connect

Engage

DESCRIPTION

Hello to all the members of the team!

We hope that you have successfully learned how to work with memory and pointers, how to create programs and implement advanced algorithms. You will need this knowledge in this Race.

Cryptography is used to secure and protect data during communication. It is helpful to prevent an unauthorized person or group of users from accessing any confidential data. Encryption and decryption are the two essential features of cryptography just like a Yin and Yang, alpha and omega, day and night, H&M.

To be able to protect yourself from hackers you have to think like a hacker. This challenge invites you to investigate the decryption process.

It might sound complicated, but it's actually very cool.
So, organize your work space and let's do it!

BIG IDEA

Data encryption.

ESSENTIAL QUESTION

How to protect confidential data?

CHALLENGE

Develop decryption algorithm.

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 to organize work productively?
- How to split work among all team members?
- What are the personal qualities of a good team leader?
- How to develop an appropriate solution algorithm?
- How to allocate memory dynamically?
- How to free memory?
- What are memory leaks? Why is everybody afraid of leaks?
- Are you sure you understand everything about Makefile? Does Oracle agree with you?
- What is the difference between an operand and an operation?
- What does a math equation consist of?
- What is the difference between encryption and decryption?
- Why use encryption and decryption?

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.

- Meet with your team. Discuss how you will organize teamwork. Create a channel for a team communication.
- Read about how to be an effective teammate.
- Boost your confidence in contributing ideas to the challenge.
- Research what skills you need for great teamwork.
- Read the story, taking everyone's ideas on board.
- Clone your git repository that is issued on the challenge page in the LMS.
- Distribute tasks among all team members.
- Read about branching and merging in `git`. These features improve your development efficiency.
- Put yourself in your teammate's place.
- Have a cup of tea or coffee with your teammates and talk to each other about the challenge.
- Start to develop the solution. Offer improvements. Test your code.
- 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.

- Challenge has to be carried out by the entire team.
- Each team member must understand the challenge and realization, and be able to reproduce it individually.
- It is your responsibility to assemble the whole team. Phone calls, SMS, messengers are good ways to stay in touch.
- You can proceed to `Act: Creative` only after you have completed all requirements in `Act: Basic`. But before you begin to complete the challenge, pay attention to the program's architecture. Take into account the fact that many features indicated in the `Act: Creative` require special architecture. And in order not to rewrite all the code somewhen, we recommend you initially determine what exactly you will do in the future. Note that the `Act: Basic` part gives the minimum points to validate the challenge.
- 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. Memory which is no longer needed must be released otherwise the task is considered as 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: Basic

NAME

Part of the Matrix

DIRECTORY

`./`

SUBMIT

`Makefile, inc/*.[h], src/*.[c]`

ALLOWED FUNCTIONS

`free, malloc, write, exit`

BINARY

`part_of_the_matrix`

LEGEND

Hey you!
You, another one which goes to be The One.

To be a part of the Matrix you should think like the Matrix.
You need to decode each byte of the Matrix.
You need to be a Matrix.

Architects

DESCRIPTION

Create a program that decodes all valid variations of an encrypted math expression.
The program takes four arguments:

- integer operands: `operand1` and `operand2`
- operation `operation`
- result of the math operation `result`

Your encoder must support four math operations: addition `+`, subtraction `-`, multiplication `*` and division `/`. Operands and the result may consist of hidden digits marked by `?` character. One digit per one `?`. The program must not print zeros if they are before a full number.

You need to find out which digits are hidden by `?` in order to maintain a mathematical expression valid. The program must output all valid variations (one per line).
Valid expressions in output must be sorted in ascending order of `operand1`. Operands, operations, equal sign and the result in output are separated by a single `space` character.

Error handling. The program prints errors to the standard error stream `stderr`:

- if the number of arguments is not equal to 4, the program prints `usage: ./part_of_the_matrix [operand1] [operation] [operand2] [result]`
- in case of invalid operation the program prints `Invalid operation: <value>`
- in case of invalid operands the program prints `Invalid operand: <value>`
- in case of invalid operation the program prints `Invalid result: <value>`

All errors must be followed by a newline. Remember, that `cat -e` doesn't show `\n` characters in the error stream. If there are more than 1 error, the program prints only 1 error message according to the priority listed above.

In the Act: Basic the program must work only with positive values. Only positive values makes sense. Find examples in the CONSOLE OUTPUT.

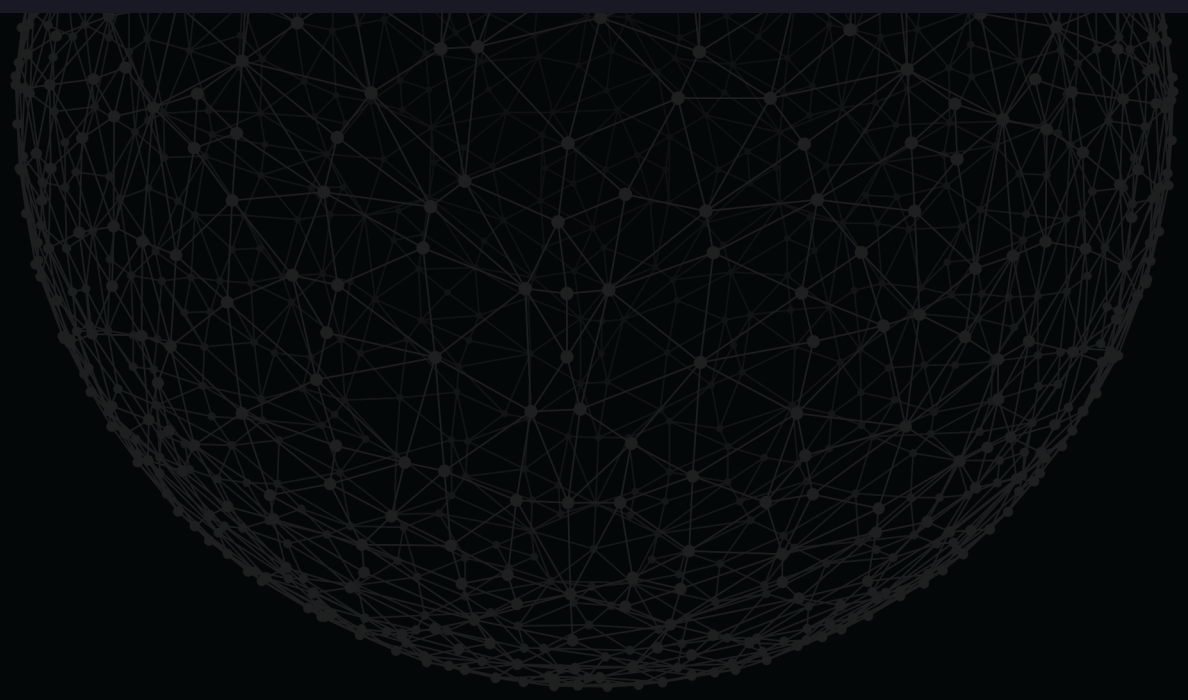
P.S: when we talk about division - we are using `quotient` in `result` in case of Euclidean division.

Make sure that you've read the section about Makefile in the Auditor really carefully.

CONSOLE OUTPUT

```
>./part_of_the_matrix | cat -e
usage: ./part_of_the_matrix [operand1] [operation] [operand2] [result]
>./part_of_the_matrix "xxx" "+" "?0?" "1?0" | cat -e
Invalid operand: xxx
>./part_of_the_matrix "xxx" "??" "?0?" "1?0" | cat -e
Invalid operation: ??
>./part_of_the_matrix "???" "??" "XXX" "1?0" | cat -e
Invalid operation: ??
>./part_of_the_matrix "???" "-" "100" "?" | cat -e
100 - 100 = 0$
101 - 100 = 1$
102 - 100 = 2$
103 - 100 = 3$
104 - 100 = 4$
105 - 100 = 5$
106 - 100 = 6$
107 - 100 = 7$
108 - 100 = 8$
109 - 100 = 9$
>./part_of_the_matrix "343111 " "      - " "??????" "    32410" | cat -e
343111 - 310701 = 32410$
>./part_of_the_matrix "?" "*" "11" "??" | cat -e
0 * 11 = 0$
1 * 11 = 11$
```

```
2 * 11 = 22$
3 * 11 = 33$
4 * 11 = 44$
5 * 11 = 55$
6 * 11 = 66$
7 * 11 = 77$
8 * 11 = 88$
9 * 11 = 99$
>./part_of_the_matrix "  ??  " "  /  " "  11  " "  1  " | cat -e
11 / 11 = 1$
12 / 11 = 1$
13 / 11 = 1$
14 / 11 = 1$
15 / 11 = 1$
16 / 11 = 1$
17 / 11 = 1$
18 / 11 = 1$
19 / 11 = 1$
20 / 11 = 1$
21 / 11 = 1$
>./part_of_the_matrix "69" "/" "0" "?" | cat -e
>
```



Act: Creative

DESCRIPTION

Handle:

- a mix of `digits` and `?` characters in operands and result
- `?` in `operation`. Furthermore, you need to sort valid expressions in the order of `operation (+, -, *, /)` and then by `operand1` in output
- `negative` numbers in `operand1`, `operand2` and `result`

Find an example in the CONSOLE OUTPUT.

CONSOLE OUTPUT

```
>./part_of_the_matrix "?11" "+" "?0?" "1?0" | cat -e
11 + 109 = 120$
111 + 9 = 120$
>./part_of_the_matrix "?" "?" "1" "8" | cat -e
7 + 1 = 8$
9 - 1 = 8$
8 * 1 = 8$
8 / 1 = 8$
>./part_of_the_matrix "-3" "+" "-?4" "-2?" | cat -e
-3 + -24 = -27$
>./part_of_the_matrix "-2" "?" "?2" "-2?" | cat -e
-2 - 22 = -24$
-2 * 12 = -24$
>
```

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.