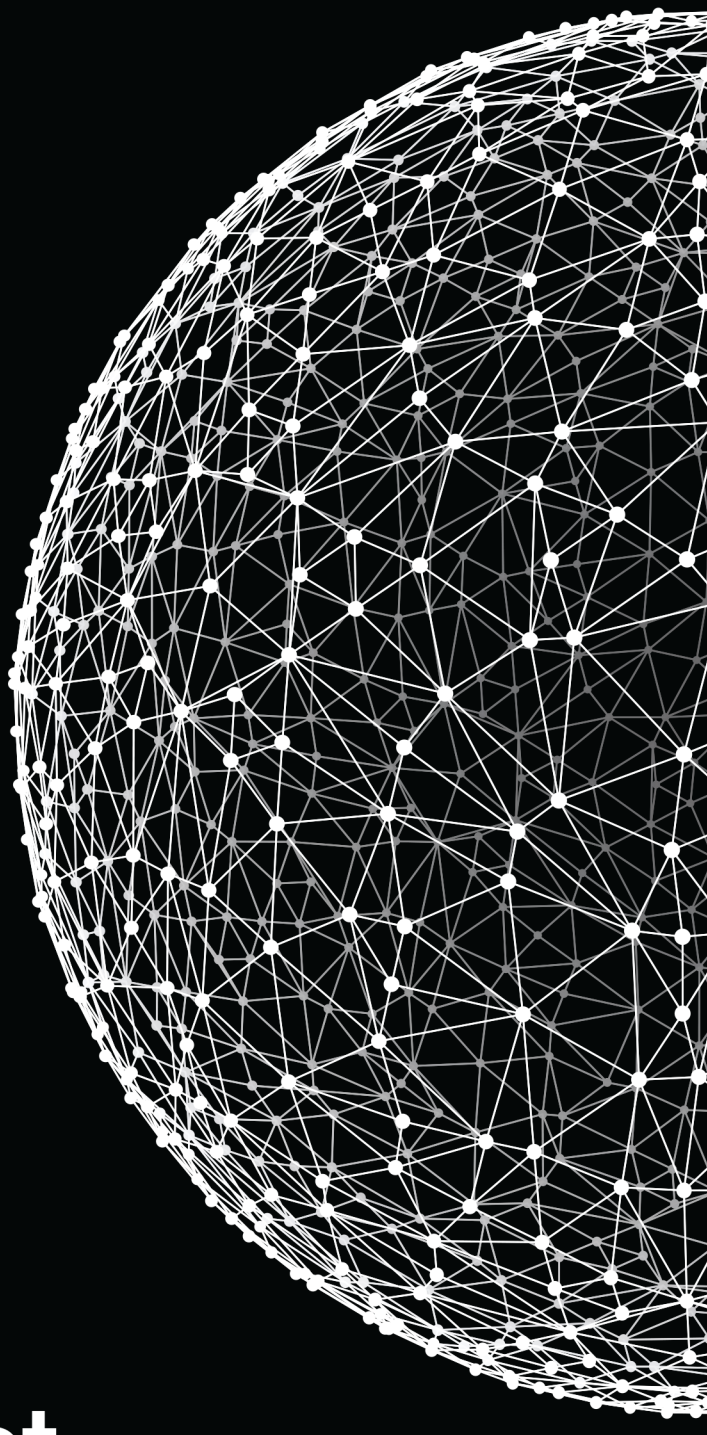
Sprint 00

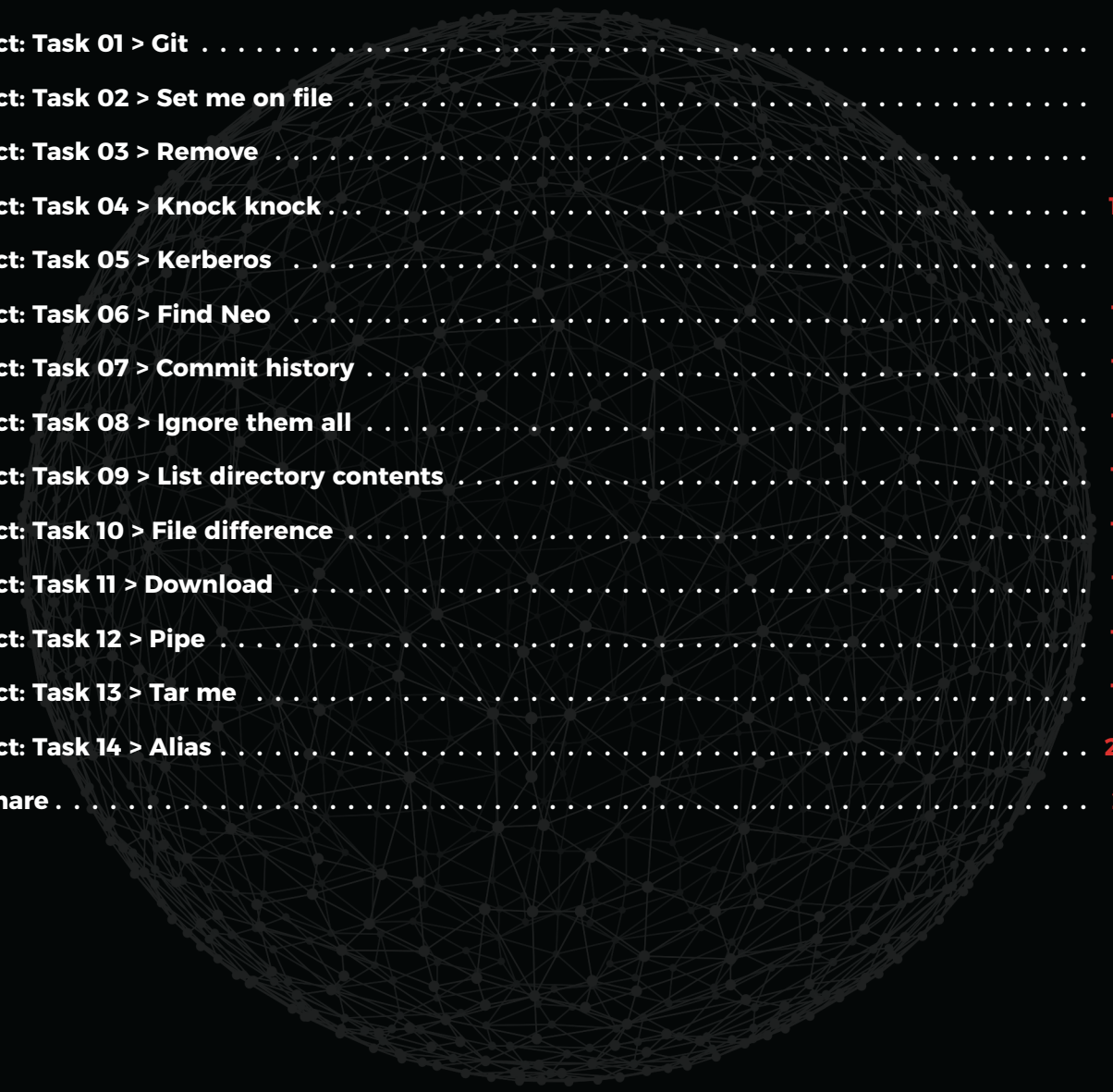
Marathon C

September 3, 2020



ucode connect

Contents



ucode connect

Engage

DESCRIPTION

Hey there!

You code world, and it's time to prove it. During this Marathon, you will overcome a lot of challenges. With each challenge you overcome, you will gain skills that will be useful to you in life in any situation and under any circumstances.

During the Marathon C, you will get a strong knowledge base of computer science. It will be difficult, but it will be worth it. After completing this Marathon, you'll be ready to proceed to the other challenges, technologies and programming languages.

Well then, no time to waste, let's get started.

And remember, education is not preparation for life.
Education is a part of life.

BIG IDEA

Find your way to success.

ESSENTIAL QUESTION

How to effectively use all the components of the educational system to get as much experience as possible?

CHALLENGE

Start learning programming.

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.

- What is your name? How old are you? What do you do in life?
- What do you know about programming?
- What is your background in programming?
- Why are you interested in learning programming?
- What ideas can be implemented using programming?
- What skills do you want to get?
- What product would you like to create using acquired skills?
- What do you need to start learning?
- Are you ready to start?

GUIDING ACTIVITIES

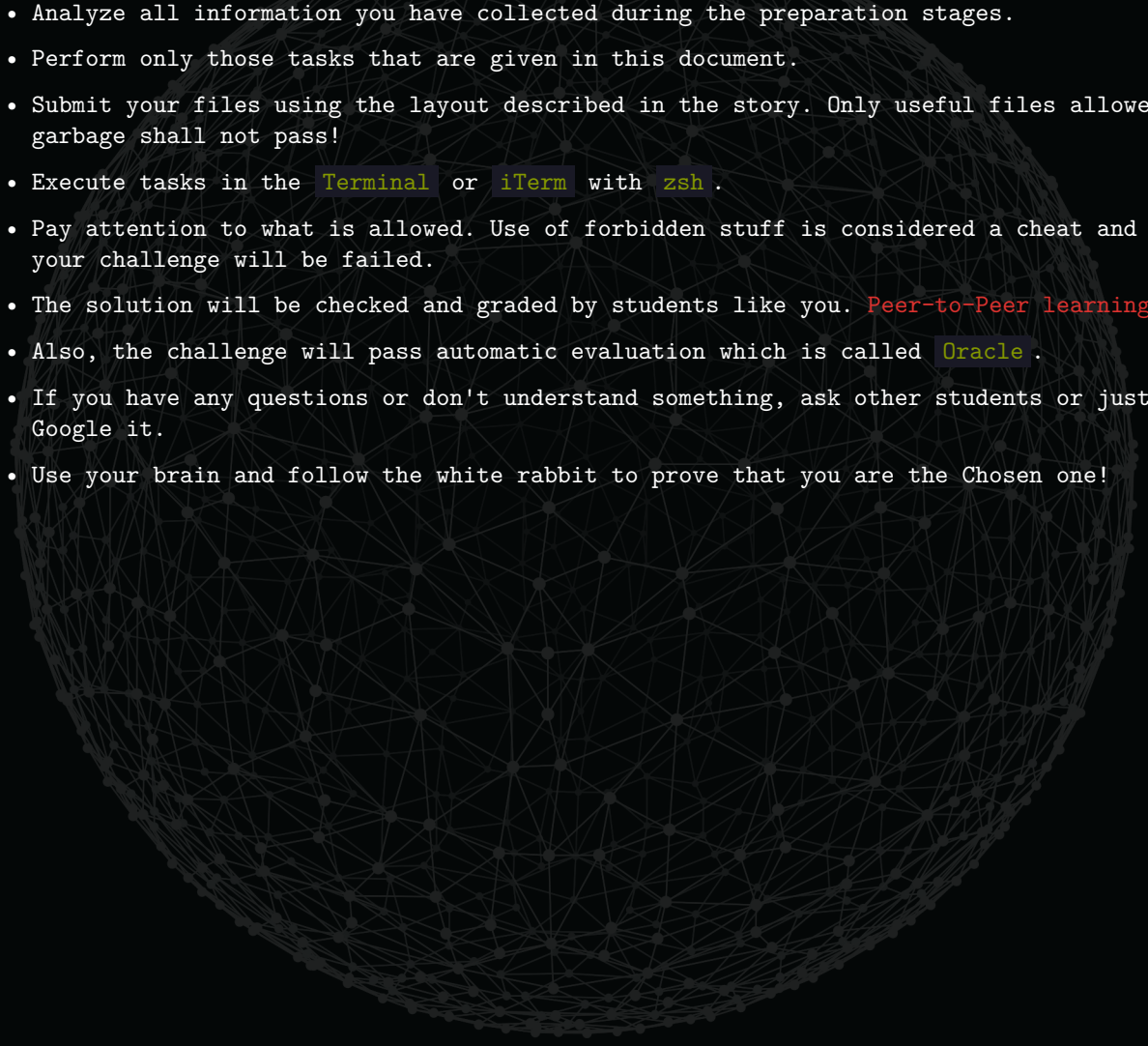
Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Get to know and understand the operating system (OS), add the languages you need.
- Read the book `ftp://ftp.oeaw.ac.at/pc/e-books/linux/learn_unix.pdf`.
- Connect to the ucode connect iMac.
- Open the `Terminal` or the `iTerm` utility. Watch the `video tutorial` about command-line interfaces.
- Type `vim`. The Vim text editor should open. Learn how to exit the editor without closing the Terminal and try opening Vim again. If you don't know how to do this - google `how to exit Vim` or watch the `video tutorial`. And now type `emacs`. Understand how to do the same actions with this text editor.
- You can find out that `Vim` and `Emacs` are directly in the Terminal. Just type `man vim` and later repeat with `man emacs`. Press `Q` to quit from `man`.
- Watch the `video tutorial` about `git` that we have prepared for you.
  You can also find it in `LMS->Media->git.mp4`
- Clone your git repository that is issued on the challenge page in the LMS.
  Use `git clone` for this.
- Proceed to the next tasks.
- 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!
- Execute tasks in the `Terminal` or `iTerm` with `zsh`.
- Pay attention to what is allowed. Use of forbidden stuff is considered a cheat and your challenge will be failed.
- 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

Man

DIRECTORY

`t00/`

SUBMIT

`man.sh`

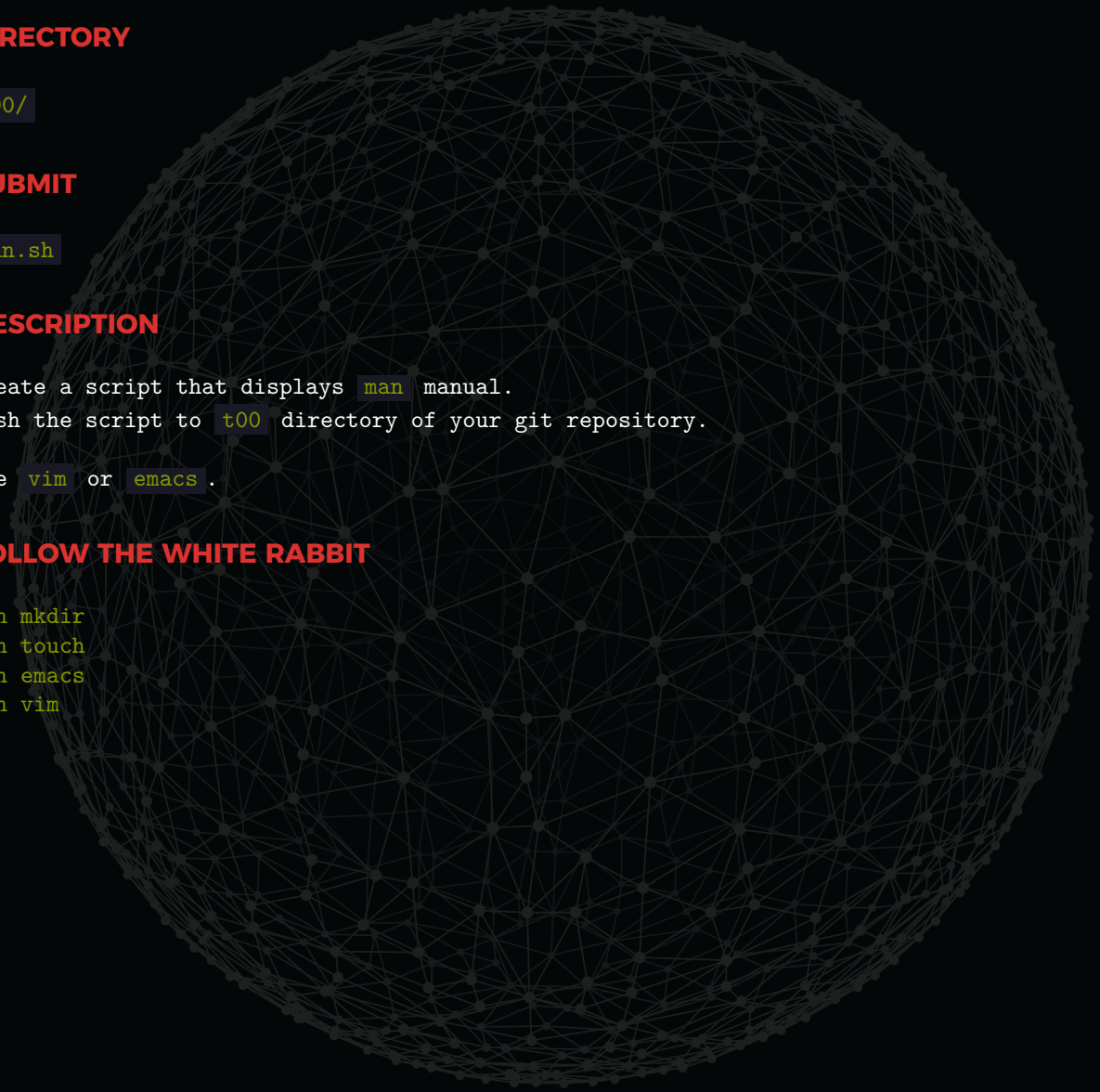
DESCRIPTION

Create a script that displays `man` manual.
Push the script to `t00` directory of your git repository.

Use `vim` or `emacs`.

FOLLOW THE WHITE RABBIT

```
man mkdir
man touch
man emacs
man vim
```



Act: Task 01

NAME

Git

DIRECTORY

`t01/`

SUBMIT

`push_me.txt`

DESCRIPTION

Create a `txt` file that contains three `git` commands that you will use to commit and push task solutions to your repository:

- add changes staged for the next commit
- commit changes with a descriptive commit message
- push committed changes to a remote repository

Each git command must be followed by a newline.

CONSOLE OUTPUT

```
>cat -e push_me.txt
git command1$
git command2$
git command3$
```

FOLLOW THE WHITE RABBIT

```
man git
man git-add
man git-commit
man git-push
```

SEE ALSO

Git user manual
Git workflow
Git Command Explorer

Act: Task 02

NAME

Set me on file

DIRECTORY

`t02/`

SUBMIT

`set_me_on_file.sh`

DESCRIPTION

Create a script that:

- creates a file called `fire`
- sets permissions and last-modified date for the created `fire` file, exactly like in the CONSOLE OUTPUT section

CONSOLE OUTPUT

```
>zsh set_me_on_file.sh
>ls -laT
total 8
drwxr-xr-x   4 xlogin  users  128 Jan  3 13:42:37 2019 .
drwxr-xr-x  19 xlogin  users  608 Jan  3 13:42:02 2019 ..
-r--------   1 xlogin  users    0 Aug 24 00:00:00 1991 fire
-rw-r--r--   1 xlogin  users   31 Jan  3 13:42:37 2019 set_me_on_file.sh
```

FOLLOW THE WHITE RABBIT

```
man chmod
man ls
```

Act: Task 03

NAME

Remove

DIRECTORY

`t03/`

SUBMIT

`remove_me.sh`

DESCRIPTION

Create a script that removes specified directories and/or files.

CONSOLE OUTPUT

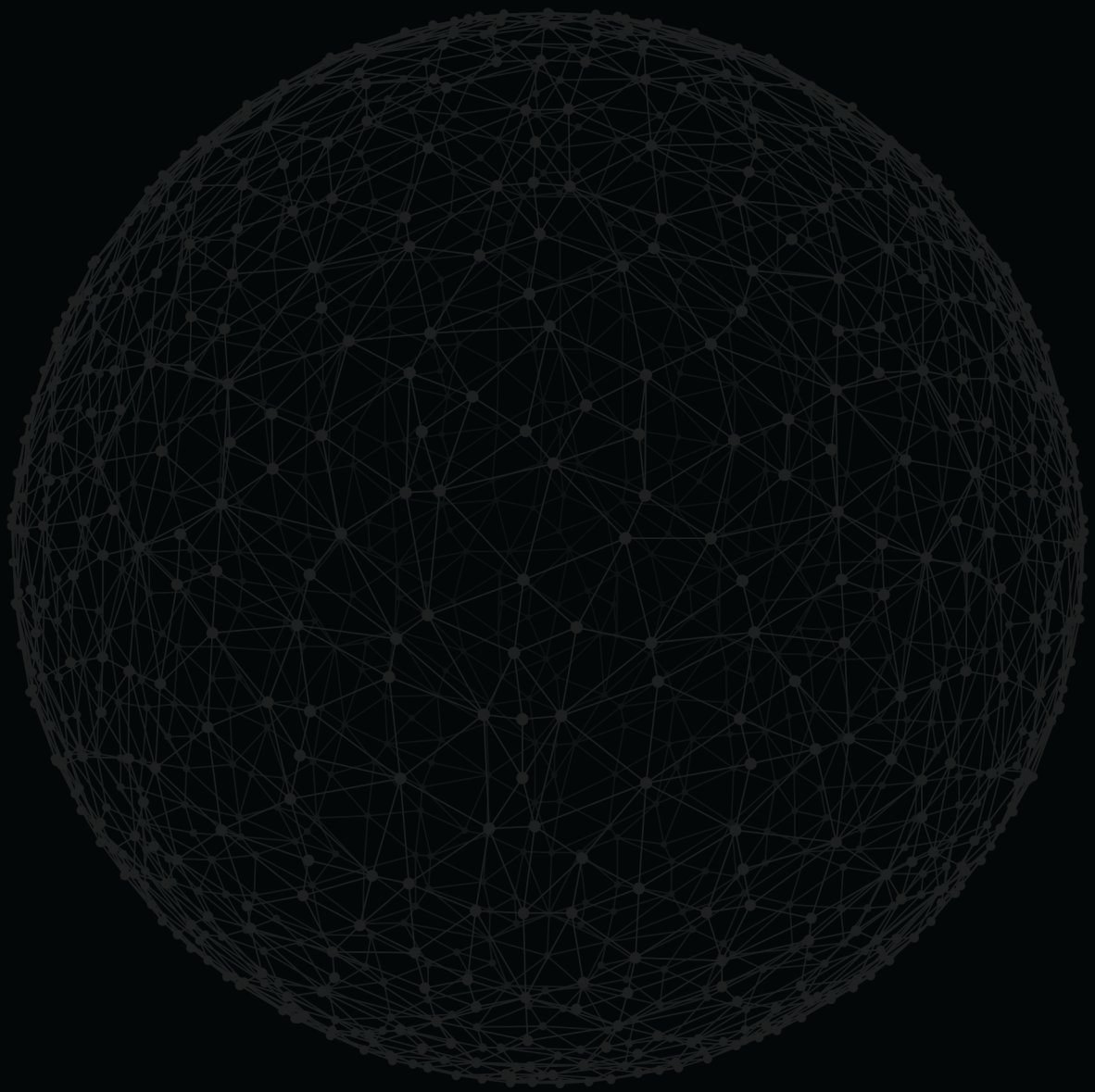
```
>ls -R
dir1   dir2   dir3   file1   file2   remove_me.sh

./dir1:

./dir2:

./dir3:
file3
>zsh remove_me.sh dir1 file1
>ls
dir2   dir3   file2   remove_me.sh
>zsh remove_me.sh dir2
>ls
dir3   file2   remove_me.sh
>zsh remove_me.sh dir2
>ls
dir3   file2   remove_me.sh
>zsh remove_me.sh dir3
rm: dir3: Directory not empty
>cd dir3
>zsh ../remove_me.sh file3
>cd ..
>zsh remove_me.sh dir3
>ls
file2   remove_me.sh
>
```

FOLLOW THE WHITE RABBIT

```
man rm
```

SEE ALSO

How to Remove Files and Directories
how to pass all parameters



Act: Task 04

NAME

Knock knock . . .

DIRECTORY

`t04/`

SUBMIT

`wake_up.sh`

DESCRIPTION

Create a script that:

- creates a file `instructions.txt`
- writes `Follow the white rabbit.` followed by a newline to `instructions.txt`

CONSOLE OUTPUT

```
>zsh wake_up.sh
>cat -e instructions.txt
Follow the white rabbit.$
>
```

Act: Task 05

NAME

Kerberos

DIRECTORY

`t05/`

SUBMIT

`kerberos.txt`

DESCRIPTION

Create a file `kerberos.txt` that contains three commands:

- display the current tickets in the credential cache
- authenticate yourself to the Kerberos server as principal on any computer
- remove all credential caches

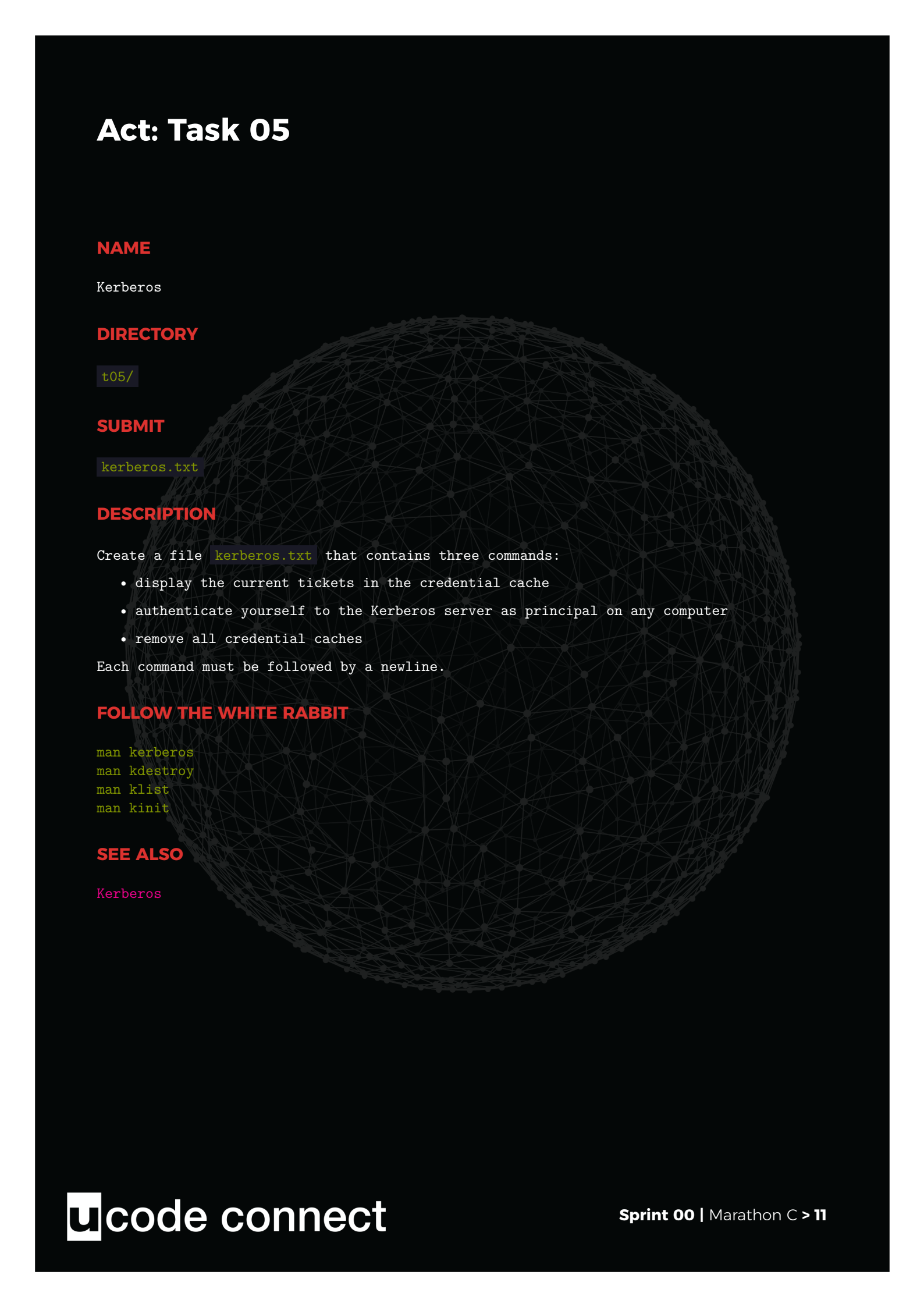
Each command must be followed by a newline.

FOLLOW THE WHITE RABBIT

```
man kerberos
man kdestroy
man klist
man kinit
```

SEE ALSO

Kerberos



Act: Task 06

NAME

Find Neo

DIRECTORY

`t06/`

SUBMIT

`find_chosen.sh`

DESCRIPTION

Create a script that:

- takes a file as an argument. The file contains Matrix characters formatted as in the CONSOLE OUTPUT section
- shows only `redpill` entities from file. Be sure to pay attention to the script's work in the CONSOLE OUTPUT

CONSOLE OUTPUT

```
>cat -e characters
Agent #0 strength:8 power:5$
Agent #1 strength:5 power:5$
Redpill Anderson strength:6 power:8$
Agent #2 strength:3 power:6$
Ageredpillnt Dozer strength:2 power:4$
redpill Dozer strength:2 power:4$
reDPill Trinity strength:9 power:9$
>zsh find_chosen.sh characters | cat -e
Redpill Anderson strength:6 power:8$
redpill Dozer strength:2 power:4$
reDPill Trinity strength:9 power:9$
>
```

Act: Task 07

NAME

Commit history

DIRECTORY

`t07/`

SUBMIT

`git_log.sh`

DESCRIPTION

Create a script that:

- shows abbreviated commit hash and subject separated by a space of three last commits

Create more than three commits in `Sprint00` repository so that the assessor will be able to check the script correctness during the defence.
Every commit must be followed by a newline. Look an example of how it can work in the CONSOLE OUTPUT.

CONSOLE OUTPUT

```
>zsh git_log.sh > git_history.txt
>cat -e git_history.txt
f61fde9 t05 find chosen$
50ab5e5 t04 kerberos$
dcf793c t03 wake up$
>
```

FOLLOW THE WHITE RABBIT

```
man git-log
man git
```

Act: Task 08

NAME

Ignore them all

DIRECTORY

`t08/`

SUBMIT

`.gitignore`

DESCRIPTION

Create a `.gitignore` for next files:

- .DS_Store
- ._.DS_Store
- *.o
- *.out

Hint: use it in your challenge repository.

CONSOLE OUTPUT

```
>touch .DS_Store && touch kek.o
>git ls-files --ignored --exclude-standard --others | cat -e
.DS_Store$
kek.o$
>
```

Act: Task 09

NAME

List directory contents

DIRECTORY

`t09/`

SUBMIT

`ls.sh`

DESCRIPTION

Create a script that:

- takes a file/directory as an argument
- shows all files/directories inside a given directory except for `.` and `..`
- displays their sizes separated by a single space
- sorts by file/directory name

Take into account, your script does not need to deal with files that have major and minor numbers.

CONSOLE OUTPUT

```
>zsh ls.sh . | cat -e
ls.sh 51B$
>zsh ls.sh /bin | cat -e
bash 604K$
cat 23K$
chmod 33K$
cp 28K$
csh 371K$
date 28K$
...
zsh 596K$
>
```

Act: Task 10

NAME

File difference

DIRECTORY

`t10/`

SUBMIT

`diff.sh`

DESCRIPTION

Create a script that:

- takes three files as arguments
- finds a difference between two files
- writes their `difference` to the third file

CONSOLE OUTPUT

```
>zsh diff.sh t10_1.txt t10_2.txt difference.txt
>cat -e difference.txt
6c6$
< <string>YES</string>$
---$
> <string>NO</string>$
18c18$
< <string>59</string>$
---$
> <string>69</string>$
28c28$
< <string>44</string>$
---$
> <string>46</string>$
>
```

Act: Task 11

NAME

Download

DIRECTORY

`t11/`

SUBMIT

`download.sh`

DESCRIPTION

Create a script that:

- takes a `url` and a `filename` as first and second arguments respectively
- downloads an image from the url and saves it to the file with the given filename

CONSOLE OUTPUT

```
>zsh download.sh https://lms.ucode.world/api/media/protocol_resources/sprint00/p4ujh.jpg image.jpg
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  161k  100  161k    0     0  89933      0  0:00:01  0:00:01 --:--:-- 89961
>open image.jpg
>
```

FOLLOW THE WHITE RABBIT

```
man curl
man open
```

Act: Task 12

NAME

Pipe

DIRECTORY

`t12/`

SUBMIT

`pipe.sh`

DESCRIPTION

Create a script that:

- takes a file as an argument. The file contains Matrix characters formatted as in the CONSOLE OUTPUT section
- shows only `redpill` entities from the file changed into `bluepill`. Be sure to pay attention to the script's work in the CONSOLE OUTPUT
- contains only one-line instruction

CONSOLE OUTPUT

```
>cat -e characters
Agent #0 strength:8 power:5$
Agent #1 strength:5 power:5$
Redpill Anderson strength:6 power:8$
Agent #2 strength:3 power:6$
Ageredpillnt Dozer strength:2 power:4$
redpill Dozer strength:2 power:4$
reDPill Trinity strength:7 power:8$
>zsh pipe.sh characters | cat -e
bluepill Anderson strength:6 power:8$
bluepill Dozer strength:2 power:4$
bluepill Trinity strength:7 power:8$
>
```

SEE ALSO

Pipelines

Act: Task 13

NAME

Tar me

DIRECTORY

`t13/`

SUBMIT

`tar.sh`

DESCRIPTION

Create a script that:

- creates a new archive with the given archive name and file/directory set
  `tar.sh [-c] [name.tar] [file ...]`
- extracts files from the given archive `tar.sh [-e] [name.tar]`

CONSOLE OUTPUT

```
>ls
dir1   dir2   file1  file2  tar.sh
>zsh tar.sh -c arch.tar dir1 dir2 file1 file2
>ls
arch.tar dir1     dir2     file1    file2    tar.sh
>rm -df dir1 dir2 file1 file2
>ls
arch.tar tar.sh
>zsh tar.sh -e arch.tar
>ls
arch.tar dir1     dir2     file1    file2    tar.sh
>
```

FOLLOW THE WHITE RABBIT

`man tar`

Act: Task 14

NAME

Alias

DIRECTORY

`t14/`

SUBMIT

`alias.sh`

DESCRIPTION

Create a script `alias.sh` that takes a filename as a command-line argument and writes in this file the aliases listed below:

- `ga` for `git add` command
- `gcmsg` for `git commit -m` command
- `gp` for `git push` command

Also, take into account that:

- aliases in the file must be in the same order as above
- each alias in the file must be followed by a newline

Find out how to put these aliases to work in your Terminal permanently to use during the entire Marathon C.

CONSOLE OUTPUT

```
>ls
alias.sh
>ga
zsh: command not found: ga
>zsh alias.sh source_me
>ls
alias.sh  source_me
>source source_me
>ga
Nothing specified, nothing added.
Maybe you wanted to say 'git add .'?
>gcmsg
error: switch `m' requires a value
>
```

SEE ALSO

Unix alias command

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.