0x00. Shell, navigation

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Create me!

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Access your sandbox and:

- Change your working directory to `/root`
- Then, create an empty file `so_cool`

Advices:

- Don’t forget to validate your current working directory
- Don’t forget to display the list of files of your current directory to validate the creation of the new file

Done! Help Check your code Get a sandbox QA Review

1. More of me

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Access your sandbox and:

- Change your working directory to `/root`
- Then, copy the file `school` to `/tmp`

Advices:

- Don’t forget to validate your current working directory
- Don’t forget to display the list of files of your current directory to validate the copy of the file

Done! Help Check your code Get a sandbox QA Review

2. To old

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Access your sandbox and:

- Change your working directory to `/root`
- Then, rename the file `old_school` to `new_school` (in the same directory)

Advices:

- Don’t forget to validate your current working directory
- Don’t forget to display the list of files of your current directory to validate the renaming of the file

Done! Help Check your code Get a sandbox QA Review

3. Not here

mandatory

Score: 66.67% (*Checks completed: 66.67%*)

Access your sandbox and:

- Change your working directory to `/root`
- Then, move the file `not_here` to `/tmp/right_school`

Advices:

- Don’t forget to validate your current working directory
- Don’t forget to display the list of files of your current directory to validate the move of the file

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

4. Not anymore

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Access your sandbox and:

- Change your working directory to `/root`
- Then, delete the file `ready_to_be_removed`

Advices:

- Don’t forget to validate your current working directory
- Don’t forget to display the list of files of your current directory to validate the removal of the file

Done! Help Check your code Get a sandbox QA Review

5. Organization is key!

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Access your sandbox and:

- Change your working directory to `/root`
- Then, create a directory `school_is_amazing`

Advices:

- Don’t forget to validate your current working directory

- Don’t forget to display the list of directories of your current directory to validate the creation of the directory

Done! Help Check your code Get a sandbox QA Review

6. No need

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Access your sandbox and:

- Change your working directory to `/root`
- Then, remove the directory `empty_directory`

Advices:

- Don’t forget to validate your current working directory
- Don’t forget to display the list of directories of your current directory to validate the removal of the directory

0x01. Emacs

ShellEditor

- By: Julien Barbier
- Weight: 1
- Project over - took place from Nov 28, 2022 6:00 AM to Nov 29, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 8.0/8 mandatory
- **Altogether: 100.0%**
  - Mandatory: 100.0%
  - Optional: no optional tasks

Concepts

For this project, we expect you to look at these concepts:

- Shell
- Using Emacs as editor
- The Framework

Congrats!

Now you know how to navigate into a Unix system!

It's time to test and decide what will be your favorite text editor on your sandbox: **Emacs** or **Vi**.

During this project you will play with **Emacs**.

Resources

Read or watch:

- A Guided Tour of Emacs

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What is Emacs
- Who is Richard Stallman
- How to open and save files
- What is a buffer and how to switch from one to the other
- How to use the mark and the point to set the region

- How to cut and paste lines and regions
- How to search forward and backward
- How to invoke commands by name
- How to undo
- How to cancel half-entered commands
- How to quit Emacs

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

- All tasks **must be** done inside the sandbox `Ubuntu 20.04`
- Your sandbox **must be available at the end of this project** - the Checker will access to it at midnight for running the correction!
- The answer of a task must be in a specific file
- Each answer file must contain only the command to execute in Emacs for solving the task. Example: "What is the command to write buffer to a specified file?" -> the file should contain only `C-x C-w`

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Create your answer directory

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Navigate to `/root` and create a directory named `0x01_emacs`

Done! Help Check your code Get a sandbox QA Review

1. Opening

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to open a file from within Emacs?

Write the answer into the file `/root/0x01_emacs/opening`.

You can validate if the format of your answer is correct by displaying the file information:

```
root@hex:~# ls -l /root/0x01_emacs/opening

-rw-r--r-- 1 root root 9 Nov 11 04:34 /root/0x01_emacs/opening
```

```
root@hex:~#
```

Done! Help Check your code Get a sandbox QA Review

2. Saving

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to save a file?

Write the answer into the file `/root/0x01_emacs/saving`.

Done! Help Check your code Get a sandbox QA Review

3. Cutting

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to cut an entire line?

Write the answer into the file `/root/0x01_emacs/cutting`.

Done! Help Check your code Get a sandbox QA Review

4. Pasting

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to paste?

Write the answer into the file `/root/0x01_emacs/pasting`.

Done! Help Check your code Get a sandbox QA Review

5. Searching

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to search forward?

Write the answer into the file `/root/0x01_emacs/searching`.

Done! Help Check your code Get a sandbox QA Review

6. Undoing

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to undo?

Write the answer into the file `/root/0x01_emacs/undoing`.

Done! Help Check your code Get a sandbox QA Review

7. Quitting

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to quit Emacs?

Write the answer into the file `/root/0x01_emacs/quitting`.

[Optional] Vagrant

DevOpsVirtual machine

- By: Julien Barbier
- Weight: 1
- Project over - took place from Nov 28, 2022 6:00 AM to Dec 3, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 1.0/1 mandatory & 10.0/10 optional
- **Altogether: 200.0%**
    - Mandatory: 100.0%
    - Optional: 100.0%
    - Calculation: 100.0% + (100.0% * 100.0%) == **200.0%**

Concepts

For this project, we expect you to look at this concept:

- Using Vagrant on your personal computer

Vagrant - or - how to code in your local computer

Sandboxes are great, but you can also do your ALX assessments on your local computer - having a virtual machine (VM) is the perfect tool for that.

Let's dig into **Vagrant** today!

Also:

- This project is **100% optional**
- This project **can't be done in Sandboxes** - it can be done only in your local computer.

Resources

Read or watch:

- Virtual machine
- man uname

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What is a virtual machine
- What is Vagrant
- Who wrote Vagrant
- What is Ubuntu
- What does “Ubuntu” mean
- How to use VMs with Vagrant
- What does the command `uname` do

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- A `README.md` file at the root of the repo, containing a description of the repository
- A `README.md` file, at the root of the folder of *this* project (i.e. `0x00-vagrant`), describing what this project is about

More Info

Install `git`

If `git` is not already installed on your terminal:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install git
```

Basic usage

At the end of this project you should be able to reproduce and understand these command lines:

```
$ git clone <repo>
$ touch test
$ git add test
$ git commit -m "Initial commit"
$ git push origin main
```

Warning

This project **can’t be done in Sandboxes** - it can be done only in your local computer. Please refer to our concept pages for your operating system.

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Create and setup your Git and GitHub account

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

You will need Git for this project, you might have to install it on your computer if it’s not done yet.

- Configure your basic info (name, email) on your local machine – they will be part of your commits. Tips

On GitHub.com:

- Using the graphic interface on the website, create the repository (if it’s not done yet)
    - Description: `This is my first repository as a full-stack engineer`
    - Public repo: `zero_day`
    - No `README`, `.gitignore`, or license

On your computer, open a terminal and do the following:

- Navigate to your home directory. Tips
- Create a directory `zero_day`. Tips
- Navigate to this new directory. Tips
- Initialize git and add the remote origin
- Create a file `README.md` with Emacs (or other command line editors) and write a small Markdown text to present this project. **This file is mandatory in projects**
- Add this new file to git, commit the change with this message “My first commit” and push to the remote server / origin (Note: You will probably need to set your login/password to push to the remote server)

Good job!

You pushed your first file in your **first repository** of the **first task** of your **first School project**.

Repo:

- GitHub repository: `zero_day`
- File: `README.md`

Done! Help Check your code QA Review

1. Hello Ubuntu

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Inside the `zero_day` repo, create a new directory called `0x00-vagrant`. Add a `README.md` file to this directory.

`ssh` into your Ubuntu VM. What does the command `uname` print when you run it without any option?

Type your answer into a file in the `0x00-vagrant` directory and push it to GitHub. Name your file accordingly as shown below.

Repo:

- GitHub repository: `zero_day`
- Directory: `0x00-vagrant`
- File: `0-hello_ubuntu`

0x02. vi

ShellEditor

- By: Julien Barbier
- Weight: 1
- Project over - took place from Nov 29, 2022 6:00 AM to Nov 30, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 7.0/8 mandatory
- **Altogether: 87.5%**
  - Mandatory: 87.5%
  - Optional: no optional tasks

Another text editor

After **Emacs**, it's time to play with **Vi**.

Resources

Read or watch:

- Basic vi Commands

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What is vi
- Who is Bill Joy
- How to start and exit vi
- What are the command and insert modes, and how to switch from one to the other
- How to edit text
- How to cut and paste lines
- How to search forward and backward
- How to undo
- How to quit vi

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.

- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

- All tasks **must be** done inside the sandbox `Ubuntu 20.04`
- Your sandbox **must be available at the end of this project** - the Checker will access to it just after the deadline to run the correction!
- The answer of a task must be in a specific file
- Each answer file must contain only the command to execute in Emacs for solving the task. Example: "What is the command to quit without saving changes?" -> the file should contain only `:q!`

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Create your answer directory

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Navigate to `/root` and create a directory named `0x02_vi`

Done! Help Check your code Get a sandbox QA Review

1. Inserting

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to insert text before the cursor?

Write the answer into the file `/root/0x02_vi/inserting`.

You can validate if the format of your answer is correct by displaying the file information:

```
root@hex:~# ls -l /root/0x02_vi/inserting
-rw-r--r-- 1 root root 2 Nov 11 04:34 /root/0x02_vi/inserting
root@hex:~#
```

Done! Help Check your code Get a sandbox QA Review

2. Cutting

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to delete and cut the current line?

Write the answer into the file `/root/0x02_vi/cutting`.

Tips:

- How to Copy, Cut and Paste

Done! Help Check your code Get a sandbox QA Review

3. Pasting

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to paste the lines in the buffer into the text after the current line?

Write the answer into the file `/root/0x02_vi/pasting`.

Done! Help Check your code Get a sandbox QA Review

4. Undoing

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to undo what you just did?

Write the answer into the file `/root/0x02_vi/undoing`.

Done! Help Check your code Get a sandbox QA Review

5. Exiting

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to quit vi even though latest changes have not been saved for this vi call?

Write the answer into the file `/root/0x02_vi/exiting`.

Done! Help Check your code Get a sandbox QA Review

6. Beginning of the line

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

What is the command to move the cursor to the start of the current line?

Write the answer into the file `/root/0x02_vi/beginning_of_the_line`.

Done! Help Check your code Get a sandbox QA Review

7. End of the line

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

What is the command to move the cursor to the end of the line?

Write the answer into the file `/root/0x02_vi/end_of_the_line`.

0x03. Git

GitCode versioningGithub

- By: Guillaume
- Weight: 1
- Project over - took place from Nov 29, 2022 6:00 AM to Nov 30, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 26.0/26 mandatory & 9.25/11 optional
- **Altogether: 184.09%**
  - Mandatory: 100.0%
  - Optional: 84.09%
  - Calculation: 100.0% + (100.0% * 84.09%) == **184.09%**

Concepts

For this project, we expect you to look at these concepts:

- Right-engineering, right-documenting
- Source code management
- Git and Github cheat sheet - Everything in less than 30 seconds

Resources

Read or watch:

- Resources to learn Git
- About READMEs
- How to write a Git commit message

Resources for advanced tasks (Read only after finishing the mandatory tasks):

- Learning branching
- Effective pull requests and other good practices for teams using GitHub

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What is source code management
- What is Git
- What is GitHub
- What is the difference between Git and GitHub
- How to create a repository
- What is a README

- How to write good READMEs
- How to commit
- How to write helpful commit messages
- How to push code
- How to pull updates
- How to create a branch
- How to merge branches
- How to work as collaborators on a project
- Which files should and which files should not appear in your repo

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- A `README.md` file at the root of the `alx-zero_day` repo, containing a description of the repository
- A `README.md` file, at the root of the folder of *this* project (i.e. `0x03-git`), describing what this project is about
- **Do not use GitHub's web UI**, but the command line to perform the exercise (except for operations that can not possibly be done any other way than through the web UI). You won't be able to perform many of the task requirements on the web UI, and you should start getting used to the command line for simple tasks because many complex tasks can only be done via the command line.
- Your answer files should only contain the command, and nothing else

More Info

Basic usage

At the end of this project you should be able to reproduce and understand these command lines:

```
$ git clone <repo>

$ touch test

$ git add test

$ git commit -m "Initial commit"

$ git push origin main
```

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Create and setup your Git and GitHub account

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Step 0 - Create an account on GitHub [if you do not have one already]

You will need a GitHub account for all your projects at ALX. If you do not already have a github.com account, you can create an account for free here

Step 1 - Create a Personal Access Token on Github

To have access to your repositories and authenticate yourself, you need to create a Personal Access Token on Github.

You can follow this tutorial to create a token.

Once it’s created, you should have a token that looks like this:

Step 2 - Update your profile on the Intranet

Update your Intranet profile by adding your Github username here

If it’s not done **the Checker won’t be able to correct your work**

Step 3 - Create your first repository

Using the graphic interface on the github website, create your first repository.

- Name: `alx-zero_day`
- Description: `I'm now a ALX Student, this is my first repository as a full-stack engineer`
- Public repo
- No `README`, `.gitignore`, or license

Step 4 - Open the sandbox

On the intranet, just under the task, click on the button and `run` to start the machine.

Once the container is started, click on to open a shell where you can start work from.

Step 5 - Clone your repository

On the webterm of the sandbox, do the following:

- Clone your repository

```
root@896cf839cf9a:/# git clone https://{YOUR_PERSONAL_TOKEN}@github.com/{YO
UR_USERNAME}/alx-zero_day.git

Cloning into 'alx-zero_day'...

warning: You appear to have cloned an empty repository.
```

Replace {YOUR_PERSONAL_TOKEN} with your token from step 1

Replace {YOUR_USERNAME} with your username from step 0 and 1

Pro-Tip: On windows, use CTRL + A + V to paste in the web terminal

Step 6 - Create the README.md and push the modifications

- Navigate to this new directory. Tips

```
root@896cf839cf9a:/# cd alx-zero_day/

root@896cf839cf9a:/alx-zero_day#
```

- Create the file `README.md` with the content `My first readme`. Tips

```
root@896cf839cf9a:/alx-zero_day# echo 'My first readme' > README.md

root@896cf839cf9a:/alx-zero_day# cat README.md

My first readme
```

- Update your git identity

```
root@896cf839cf9a:/alx-pre_course# git config --global user.email "you@exam
ple.com"

root@896cf839cf9a:/alx-pre_course# git config --global user.name "Your Name
"
```

- Add this new file to git, commit the change with this message “My first commit” and push to the remote server / origin

```
root@896cf839cf9a:/alx-zero_day# git add .

root@896cf839cf9a:/alx-zero_day# git commit -m 'My first commit'

[master (root-commit) 98eef93] My first commit

 1 file changed, 1 insertion(+)

 create mode 100644 README.md

root@896cf839cf9a:/alx-zero_day# git push

Enumerating objects: 3, done.
```

```
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 212 bytes | 212.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/{YOUR_USERNAME}/alx-zero_day.git
 * [new branch]      master -> master
```

Good job!

You pushed your first file in your **first repository** of the **first task** of your **first ALX School project**.

You can now check your repository on GitHub to see if everything is good.

Repo:

- GitHub repository: `alx-zero_day`
- File: `README.md`

Done! Help Check your code Get a sandbox QA Review

1. Repo-session

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Create a new directory called `0x03-git` in your `alx-zero_day` repo.

Make sure you include a not empty `README.md` in your directory:

- at the root of your repository `alx-zero_day`
- AND in the directory `0x03-git`

And important part: **Make sure your commit and push your code to Github - otherwise the Checker will always fail.**

Repo:

- GitHub repository: `alx-zero_day`

Done! Help Check your code Get a sandbox QA Review

2. Coding fury road

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

For the moment we have an empty project directory containing only a `README.md`. It's time to code!

- Create these directories at the root of your project: `bash`, `c`, `js`
- Create these empty files:
  - `c/c_is_fun.c`
  - `js/main.js`
  - `js/index.js`
- Create a file `bash/alx` with these two lines inside: `#!/bin/bash` and `echo "ALX"`
- Create a file `bash/school` with these two lines inside: `#!/bin/bash` and `echo "School"`

- Add all these new files to git
- Commit your changes (message: “Starting to code today, so cool”) and push to the remote server

Repo:

- GitHub repository: `alx-zero_day`
- Directory: `0x03-git`
- File: `bash/alx, bash/school, c/c_is_fun.c, js/main.js, js/index.js`

Done! Help Check your code Get a sandbox QA Review

3. Collaboration is the base of a company

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

A branch is like a copy of your project. It’s used mainly for:

- adding a feature in development
- collaborating on the same project with other developers
- not breaking your entire repository
- not upsetting your co-workers

The purpose of a branch is to isolate your work from the main code base of your project and/or from your co-workers’ work.

For this project, create a branch `update_script` and in this branch:

- Create an empty file named `bash/98`
- Update `bash/alx` by replacing `echo "ALX"` with `echo "ALX School"`
- Update `bash/school` by replacing `echo "School"` with `echo "The school is open!"`
- Add and commit these changes (message: “My personal work”)
- Push this new branch Tips

Perfect! You did an amazing update in your project and it’s isolated correctly from the **main** branch.

Ho wait, your manager needs a quick fix in your project and it needs to be deployed now:

- Change branch to `main`
- Update the file `bash/alx` by replacing `echo "ALX"` with `echo "ALX School is so cool!"`
- Delete the directory `js`
- Commit your changes (message: “Hot fix”) and push to the origin

Ouf, hot fix is done!

Repo:

- GitHub repository: `alx-zero_day`
- Directory: `0x03-git`
- File: `bash/alx, bash/school, bash/98`

Done! Help Check your code Get a sandbox QA Review

4. Collaboration: be up to date

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Of course, you can also work on the same branch as your co-workers and it’s best if you keep up to date with their changes.

For this task – **and only for this task** – please update your file `README.md` in the main branch from GitHub.com. It’s the **only time** you are allowed to update and commit from GitHub interface.

After you have done that, in your terminal:

- Get all changes of the main branch locally (i.e. your `README.md` file will be updated)
- Create a new file `up_to_date` at the root of your directory and in it, write the git command line used
- Add `up_to_date` to git, commit (message: “How to be up to date in git”), and push to the origin

Repo:

- GitHub repository: `alx-zero_day`
- Directory: `0x03-git`
- File: `README.md, up_to_date`

Done! Help Check your code Get a sandbox QA Review

5. HAAA what did you do???

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Collaboration is cool, but not really when you update the same file at the same time…

To illustrate that, please merge the branch `update_script` to `main`: “Cool, all my changes will be now part of the main branch, ready to be deployed!”

HHHHHHHAAAAAAAA

```
CONFLICT (content): Merge conflict in bash/alx
```

As you can see, you have conflicts between two branches on the same file.

Your goal now is to resolve conflicts by using the version of the branch `update_script`, and push the result to the origin.

At the end, you should have all your work from the branch `update_script` (new file and two updated files) and all latest `main` commits (new files, delete folder, etc.), *without* conflicts.

Repo:

- GitHub repository: `alx-zero_day`
- Directory: `0x03-git`

Done! Help Check your code Get a sandbox QA Review

6. Never push too much

#advanced

Score: 65.0% (*Checks completed: 100.0%*)

Create a `.gitignore` file and define a rule to never push `~` files (generated by Emacs). Tips

Repo:

- GitHub repository: `alx-zero_day`
- Directory: `0x03-git`
- File: `.gitignore`

0x00. C - Hello, World

man or help:

- `gcc`
- `printf (3)`
- `puts`
- `putchar`

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- Why C programming is awesome
- Who invented C
- Who are Dennis Ritchie, Brian Kernighan and Linus Torvalds
- What happens when you type `gcc main.c`
- What is an entry point
- What is `main`
- How to print text using `printf`, `puts` and `putchar`
- How to get the size of a specific type using the unary operator `sizeof`
- How to compile using `gcc`
- What is the default program name when compiling with `gcc`
- What is the official C coding style and how to check your code with `betty-style`
- How to find the right header to include in your source code when using a standard library function
- How does the `main` function influence the return value of the program

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

C

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file at the root of the repo, containing a description of the repository

- A `README.md` file, at the root of the folder of *this* project, containing a description of the project
- There should be no errors and no warnings during compilation
- You are not allowed to use `system`
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl

Shell Scripts

- Allowed editors: `vi`, `vim`, `emacs`
- All your scripts will be tested on Ubuntu 20.04 LTS
- All your scripts should be exactly two lines long (`$ wc -l file` should print 2)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/bin/bash`

More Info

Betty linter

To run the Betty linter just with command `betty <filename>`:

- Go to the Betty repository
- Clone the repo to your local machine
- `cd` into the Betty directory
- Install the linter with `sudo ./install.sh`
- `emacs` or `vi` a new file called `betty`, and copy the script below:

```
#!/bin/bash

# Simply a wrapper script to keep you from having to use betty-style

# and betty-doc separately on every item.

# Originally by Tim Britton (@wintermanc3r), multiargument added by

# Larry Madeo (@hillmonkey)


BIN_PATH="/usr/local/bin"

BETTY_STYLE="betty-style"

BETTY_DOC="betty-doc"


if [ "$#" = "0" ]; then

    echo "No arguments passed."

    exit 1

fi
```

```
for argument in "$@" ; do
    echo -e "\n========== $argument =========="
    ${BIN_PATH}/${BETTY_STYLE} "$argument"
    ${BIN_PATH}/${BETTY_DOC} "$argument"
done
```

- Once saved, exit file and change permissions to apply to all users with `chmod a+x betty`
- Move the `betty` file into `/bin/` directory or somewhere else in your `$PATH` with `sudo mv betty /bin/`

You can now type `betty <filename>` to run the Betty linter!

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Preprocessor

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that runs a C file through the preprocessor and save the result into another file.

- The C file name will be saved in the variable `$CFILE`
- The output should be saved in the file `c`

```
julien@ubuntu:~/c/0x00$ cat main.c
#include <stdio.h>

/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
int main(void)
{
    return (0);
}
julien@ubuntu:~/c/0x00$ export CFILE=main.c
julien@ubuntu:~/c/0x00$ ./0-preprocessor
julien@ubuntu:~/c/0x00$ tail c
```

```
# 942 "/usr/include/stdio.h" 3 4


# 2 "main.c" 2




# 3 "main.c"
int main(void)
{
 return (0);
}
julien@ubuntu:~/c/0x00$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x00-hello_world`
- File: `0-preprocessor`

Done! Help Check your code Get a sandbox QA Review

1. Compiler

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that compiles a C file but does not link.

- The C file name will be saved in the variable `$CFILE`
- The output file should be named the same as the C file, but with the extension `.o` instead of `.c`.
  - Example: if the C file is `main.c`, the output file should be `main.o`

```
julien@ubuntu:~/c/0x00$ export CFILE=main.c
julien@ubuntu:~/c/0x00$ cat main.c
#include <stdio.h>


/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
int main(void)
```

```
{

    return (0);

}

julien@ubuntu:~/c/0x00$ ./1-compiler

julien@ubuntu:~/c/0x00$ ls

0-preprocessor  1-compiler   c            main.o

Makefile               100-intel      main.c  main.s

julien@ubuntu:~/c/0x00$ cat -v main.o | head

^?ELF^B^A^A^@^@^@^@^@^@^@^@^@^A^@>^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^P^B^@^@^@^@^@^@^@^@^@^@@^@^@^@^@@^@^K^@^H^@UHM-^IM-eM-8^@^@^@^@]M-C^@GC
C: (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0 20160609^@^T^@^@^@^@^@^@^AzR^@^A
x^P^A^[^L^G^HM-^P^A^@^@^\^@^@^@^\^@^@^@^@^@^@^K^@^@^@^@A^N^PM-^F^BC^M^FF^
L^G^H^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^D^@M-qM
-^?^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^C^@^B^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^C^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^E^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^C^@^F^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^D^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^H^@^@^@^R^@^A^@^@^@^@^@^@^@^@^@^K^@^@^@^@^@^@^@^@ma
in.c^@main^@^@^@^@ ^@^@^@^@^@^@^@^B^@^@^@^B^@^@^@^@^@^@^@^@^@^@^@.symtab^
@.strtab^@.shstrtab^@.text^@.data^@.bss^@.comment^@.note.GNU-stack^@.rela.e
h_frame^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^[^@^
@^@^A^@^@^@^F^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@^@^@^@^@^@^@^@^K^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@!^@^@^@^A^@^@^@^C^@^@^@^@^@^
@^@^@^@^@^@^@^@@K^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^
@^@^@^@^@^@^@^@^@^@^@'^@^@^@^H^@^@^@^C^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@K^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@,
^@^@^@^A^@^@^@0^@^@^@^@^@^@^@^@^@^@^@^@^@^@K^@^@^@^@^@^@5^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^A^@^@^@^@^@^@5^@^@^@^A^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@M-^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^
@^@^@^@^@^@^@^@^@^@^@^@J^@^@^@^A^@^@^@^B^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@M
-^@^@^@^@^@^@^@8^@^@^@^@^@^@^@^@^@^@^@^@^@^@^H^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@E^@^@^@^D^@^@^@@^@^@^@^@^@^@^@^@^@^@^@^@^@M- ^A^@^@^@^@^@^@^X^@^@^
@^@^@^@^@    ^@^@^@^F^@^@^@^H^@^@^@^@^@^@^X^@^@^@^@^@^@^Q^@^@^@^C^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@M-8^A^@^@^@^@^@^@T^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^B^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@M-8^@^@^@^@^@^@^@M-X^@^@^@^@^@^@^@

^@^@^@^H^@^@^@^H^@^@^@^@^@^@^X^@^@^@^@^@^@  ^@^@^@^C^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@M-^P^A^@^@^@^@^@^@^M^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@
^@^@^@^@^@^@^@^@^@^@^@^@julien@ubuntu:~/c/0x00$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x00-hello_world`
- File: `1-compiler`

Done! Help Check your code Get a sandbox QA Review

2. Assembler

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that generates the assembly code of a C code and save it in an output file.

- The C file name will be saved in the variable `$CFILE`
- The output file should be named the same as the C file, but with the extension `.s` instead of `.c`.
    - Example: if the C file is `main.c`, the output file should be `main.s`

```
julien@ubuntu:~/c/0x00$ export CFILE=main.c
julien@ubuntu:~/c/0x00$ cat main.c
#include <stdio.h>

/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
int main(void)
{
    return (0);
}
julien@ubuntu:~/c/0x00$ ./2-assembler
julien@ubuntu:~/c/0x00$ ls
0-preprocessor  1-compiler  2-assembler c  main.c  main.s  Makefile
julien@ubuntu:~/c/0x00$ cat main.s
    .file   "main.c"
    .text
    .globl  main
    .type   main, @function
main:
.LFB0:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    movl    $0, %eax
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE0:
    .size   main, .-main
    .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0 20160609"
    .section    .note.GNU-stack,"",@progbits
julien@ubuntu:~/c/0x00$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x00-hello_world`
- File: `2-assembler`

Done! Help Check your code Get a sandbox QA Review

3. Name

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that compiles a C file and creates an executable named `cisfun`.

- The C file name will be saved in the variable `$CFILE`

```
julien@ubuntu:~/c/0x00$ export CFILE=main.c
julien@ubuntu:~/c/0x00$ cat main.c
#include <stdio.h>

/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
```

```
int main(void)
{
    return (0);
}
julien@ubuntu:~/c/0x00$ ./3-name
julien@ubuntu:~/c/0x00$ ls
0-preprocessor  1-compiler   3-name  cisfun  main.o  Makefile
100-intel       2-assembler  c       main.c  main.s
julien@ubuntu:~/c/0x00$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x00-hello_world`
- File: `3-name`

Done! Help Check your code Get a sandbox QA Review

4. Hello, puts

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a C program that prints exactly `"Programming is like building a multilingual puzzle`, followed by a new line.

- Use the function `puts`
- You are not allowed to use `printf`
- Your program should end with the value `0`

```
julien@ubuntu:~/c/0x00$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 4-pu
ts.c && ./a.out
"Programming is like building a multilingual puzzle
julien@ubuntu:~/c/0x00$ echo $?
0
julien@ubuntu:~/c/0x00$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x00-hello_world`
- File: `4-puts.c`

Done! Help Check your code Get a sandbox QA Review

5. Hello, printf

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a C program that prints exactly `with proper grammar, but the outcome is a piece of art,`, followed by a new line.

- Use the function `printf`
- You are not allowed to use the function `puts`
- Your program should return `0`
- Your program should compile without warning when using the `-Wall` `gcc` option

```
julien@ubuntu:~/c/0x00$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 5-printf.c
julien@ubuntu:~/c/0x00$ ./a.out
with proper grammar, but the outcome is a piece of art,
julien@ubuntu:~/c/0x00$ echo $?
0
julien@ubuntu:~/c/0x00$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x00-hello_world`
- File: `5-printf.c`

Done! Help Check your code Get a sandbox QA Review

6. Size is not grandeur, and territory does not make a nation

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a C program that prints the size of various types on the computer it is compiled and run on.

- You should produce the exact same output as in the example
- Warnings are allowed
- Your program should return `0`
- You might have to install the package `libc6-dev-i386` on your Linux to test the `-m32` `gcc` option

```
julien@ubuntu:~/c/0x00$ gcc 6-size.c -m32 -o size32 2> /tmp/32
julien@ubuntu:~/c/0x00$ gcc 6-size.c -m64 -o size64 2> /tmp/64
julien@ubuntu:~/c/0x00$ ./size32
Size of a char: 1 byte(s)
Size of an int: 4 byte(s)
Size of a long int: 4 byte(s)
Size of a long long int: 8 byte(s)
Size of a float: 4 byte(s)
julien@ubuntu:~/c/0x00$ ./size64
Size of a char: 1 byte(s)
Size of an int: 4 byte(s)
Size of a long int: 8 byte(s)
Size of a long long int: 8 byte(s)
Size of a float: 4 byte(s)
julien@ubuntu:~/c/0x00$ echo $?
0
julien@ubuntu:~/c/0x00$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x00-hello_world`
- File: `6-size.c`

Done! Help Check your code Get a sandbox QA Review

7. Intel

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that generates the assembly code (Intel syntax) of a C code and save it in an output file.

- The C file name will be saved in the variable `$CFILE`.
- The output file should be named the same as the C file, but with the extension `.s` instead of `.c`.
  - Example: if the C file is `main.c`, the output file should be `main.s`

```
julien@ubuntu:~/c/0x00$ export CFILE=main.c
julien@ubuntu:~/c/0x00$ cat main.c
#include <stdio.h>


/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
int main(void)
```

```
{
    return (0);
}
julien@ubuntu:~/c/0x00$ ./100-intel
julien@ubuntu:~/c/0x00$ cat main.s
    .file   "main.c"
    .intel_syntax noprefix
    .text
    .globl  main
    .type   main, @function
main:
.LFB0:
    .cfi_startproc
    push    rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    mov rbp, rsp
    .cfi_def_cfa_register 6
    mov eax, 0
    pop rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE0:
    .size   main, .-main
    .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0 20160609"
    .section    .note.GNU-stack,"",@progbits
julien@ubuntu:~/c/0x00$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x00-hello_world`
- File: `100-intel`

Done! Help Check your code Get a sandbox QA Review

8. UNIX is basically a simple operating system, but you have to be a genius to understand the simplicity

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a C program that prints exactly `and that piece of art is useful" - Dora Korpar, 2015-10-19`, followed by a new line, to the standard error.

- You are not allowed to use any functions listed in the NAME section of the man (3) `printf` or man (3) `puts`
- Your program should return 1
- Your program should compile without any warnings when using the `-Wall` `gcc` option

```
julien@ubuntu:~/c/0x00$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 -o q
uote 101-quote.c

julien@ubuntu:~/c/0x00$ ./quote

and that piece of art is useful" - Dora Korpar, 2015-10-19

julien@ubuntu:~/c/0x00$ echo $?

1

julien@ubuntu:~/c/0x00$ ./quote 2> q

julien@ubuntu:~/c/0x00$ cat q

and that piece of art is useful" - Dora Korpar, 2015-10-19

julien@ubuntu:~/c/0x00$ grep printf < 101-quote.c

julien@ubuntu:~/c/0x00$ grep put < 101-quote.c

julien@ubuntu:~/c/0x00$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x00-hello_world`
- File: `101-quote.c`

0x01. C - Variables, if, else, while

man or help:

- `ascii` (*You do not need to learn about `scanf`, `getc`, `getchar`, `EOF`, `EXIT_SUCCESS`, `time`, `rand`, `srand`, `RAND_MAX`, `for` loops, `do...while` loops, functions.*)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are the arithmetic operators and how to use them
- What are the logical operators (sometimes called boolean operators) and how to use them
- What the the relational operators and how to use them
- What values are considered TRUE and FALSE in C
- What are the boolean operators and how to use them
- How to use the `if`, `if ... else` statements
- How to use comments
- How to declare variables of types `char`, `int`, `unsigned int`
- How to assign values to variables
- How to print the values of variables of type `char`, `int`, `unsigned int` with `printf`
- How to use the `while` loop
- How to use variables with the `while` loop
- How to print variables using `printf`
- What is the `ASCII` character set
- What are the purpose of the `gcc` flags `-m32` and `-m64`

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line

- A `README.md` file, at the root of the folder of the project
- There should be no errors and no warnings during compilation
- You are not allowed to use `system`
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Positive anything is better than negative nothing

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

This program will assign a random number to the variable `n` each time it is executed. Complete the source code in order to print whether the number stored in the variable `n` is positive or negative.

- You can find the source code here
- The variable `n` will store a different value every time you will run this program
- You don't have to understand what `rand`, `srand`, `RAND_MAX` do. Please do not touch this code
- The output of the program should be:
  - The number, followed by
    - if the number is greater than 0: `is positive`
    - if the number is 0: `is zero`
    - if the number is less than 0: `is negative`
  - followed by a new line

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-positive_or_negative.c -o 0-positive_or_negative
julien@ubuntu:~/0x01$ ./0-positive_or_negative
-520693284 is negative
julien@ubuntu:~/0x01$ ./0-positive_or_negative
-973398895 is negative
julien@ubuntu:~/0x01$ ./0-positive_or_negative
-199220452 is negative
julien@ubuntu:~/0x01$ ./0-positive_or_negative
561319348 is positive
julien@ubuntu:~/0x01$ ./0-positive_or_negative
561319348 is positive
julien@ubuntu:~/0x01$ ./0-positive_or_negative
266853958 is positive
```

```
julien@ubuntu:~/0x01$ ./0-positive_or_negative

-48147767 is negative

julien@ubuntu:~/0x01$ ./0-positive_or_negative

0 is zero

julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `0-positive_or_negative.c`

Done! Help Check your code Get a sandbox QA Review

1. The last digit

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

This program will assign a random number to the variable `n` each time it is executed. Complete the source code in order to print the last digit of the number stored in the variable `n`.

- You can find the source code here
- The variable `n` will store a different value every time you run this program
- You don't have to understand what `rand`, `srand`, and `RAND_MAX` do. Please do not touch this code
- The output of the program should be:
  - The string `Last digit of`, followed by
  - `n`, followed by
  - the string `is`, followed by
    - if the last digit of `n` is greater than 5: the string `and is greater than 5`
    - if the last digit of `n` is 0: the string `and is 0`
    - if the last digit of `n` is less than 6 and not 0: the string `and is less than 6 and not 0`
  - followed by a new line

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-last
_digit.c -o 1-last_digit

julien@ubuntu:~/0x01$ ./1-last_digit

Last digit of 629438752 is 2 and is less than 6 and not 0

julien@ubuntu:~/0x01$ ./1-last_digit

Last digit of -748255693 is -3 and is less than 6 and not 0

julien@ubuntu:~/0x01$ ./1-last_digit

Last digit of -1052791662 is -2 and is less than 6 and not 0

julien@ubuntu:~/0x01$ ./1-last_digit

Last digit of -284805734 is -4 and is less than 6 and not 0
```

```
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of -284805734 is -4 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 491506926 is 6 and is greater than 5
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 954249937 is 7 and is greater than 5
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 652334952 is 2 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of -729688197 is -7 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of -729688197 is -7 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 45528266 is 6 and is greater than 5
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 45528266 is 6 and is greater than 5
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 809065140 is 0 and is 0
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `1-last_digit.c`

Done! Help Check your code Get a sandbox QA Review

2. I sometimes suffer from insomnia. And when I can't fall asleep, I play what I call the alphabet game

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints the alphabet in lowercase, followed by a new line.

- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` twice in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-prin
t_alphabet.c -o 2-print_alphabet
```

```
julien@ubuntu:~/0x01$ ./2-print_alphabet
abcdefghijklmnopqrstuvwxyz
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `2-print_alphabet.c`

Done! Help Check your code Get a sandbox QA Review

3. alphABET

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints the alphabet in lowercase, and then in uppercase, followed by a new line.

- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` three times in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-prin
t_alphabets.c -o 3-print_alphabets
julien@ubuntu:~/0x01$ ./3-print_alphabets | cat -e
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ$
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `3-print_alphabets.c`

Done! Help Check your code Get a sandbox QA Review

4. When I was having that alphabet soup, I never thought that it would pay off

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints the alphabet in lowercase, followed by a new line.

- Print all the letters except `q` and `e`
- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` twice in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-prin
t_alphabt.c -o 4-print_alphabt

julien@ubuntu:~/0x01$ ./4-print_alphabt

abcdfghijklmnoprstuvwxyz

julien@ubuntu:~/0x01$ ./4-print_alphabt | grep [eq]

julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `4-print_alphabt.c`

Done! Help Check your code Get a sandbox QA Review

5. Numbers

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints all single digit numbers of base 10 starting from `0`, followed by a new line.

- All your code should be in the `main` function

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5-prin
t_numbers.c -o 5-print_numbers

julien@ubuntu:~/0x01$ ./5-print_numbers

0123456789

julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `5-print_numbers.c`

Done! Help Check your code Get a sandbox QA Review

6. Numberz

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints all single digit numbers of base 10 starting from `0`, followed by a new line.

- You are not allowed to use any variable of type `char`
- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)

- You can only use `putchar` twice in your code
- All your code should be in the `main` function

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 6-print_numberz.c -o 6-print_numberz
julien@ubuntu:~/0x01$ ./6-print_numberz
0123456789
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `6-print_numberz.c`

Done! Help Check your code Get a sandbox QA Review

7. Smile in the mirror

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints the lowercase alphabet in reverse, followed by a new line.

- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` twice in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 7-print_tebahpla.c -o 7-print_tebahpla
julien@ubuntu:~/0x01$ ./7-print_tebahpla
zyxwvutsrqponmlkjihgfedcba
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `7-print_tebahpla.c`

Done! Help Check your code Get a sandbox QA Review

8. Hexadecimal

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints all the numbers of base 16 in lowercase, followed by a new line.

- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` three times in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 8-print_base16.c -o 8-print_base16
julien@ubuntu:~/0x01$ ./8-print_base16
0123456789abcdef
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `8-print_base16.c`

Done! Help Check your code Get a sandbox QA Review

9. Patience, persistence and perspiration make an unbeatable combination for success

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints all possible combinations of single-digit numbers.

- Numbers must be separated by `,`, followed by a space
- Numbers should be printed in ascending order
- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` four times maximum in your code
- You are not allowed to use any variable of type `char`

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 9-print_comb.c -o 9-print_comb
julien@ubuntu:~/0x01$ ./9-print_comb | cat -e
0, 1, 2, 3, 4, 5, 6, 7, 8, 9$
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `9-print_comb.c`

Done! Help Check your code Get a sandbox QA Review

10. Inventing is a combination of brains and materials. The more brains you use, the less material you need

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints all possible different combinations of two digits.

- Numbers must be separated by `,`, followed by a space
- The two digits must be different
- `01` and `10` are considered the same combination of the two digits `0` and `1`
- Print only the smallest combination of two digits
- Numbers should be printed in ascending order, with two digits
- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)
- You can only use `putchar` five times maximum in your code
- You are not allowed to use any variable of type `char`
- All your code should be in the `main` function

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-pr
int_comb3.c -o 100-print_comb3

julien@ubuntu:~/0x01$ ./100-print_comb3

01, 02, 03, 04, 05, 06, 07, 08, 09, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24,
25, 26, 27, 28, 29, 34, 35, 36, 37, 38, 39, 45, 46, 47, 48, 49, 56, 57, 58,
59, 67, 68, 69, 78, 79, 89

julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `100-print_comb3.c`

Done! Help Check your code Get a sandbox QA Review

11. The success combination in business is: Do what you do better... and: do more of what you do...

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints all possible different combinations of three digits.

- Numbers must be separated by `,`, followed by a space
- The three digits must be different
- `012`, `120`, `102`, `021`, `201`, `210` are considered the same combination of the three digits `0`, `1` and `2`
- Print only the smallest combination of three digits
- Numbers should be printed in ascending order, with three digits
- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)
- You can only use `putchar` six times maximum in your code
- You are not allowed to use any variable of type `char`
- All your code should be in the `main` function

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 101-pr
int_comb4.c -o 101-print_comb4

julien@ubuntu:~/0x01$ ./101-print_comb4

012, 013, 014, 015, 016, 017, 018, 019, 023, 024, 025, 026, 027, 028, 029,
034, 035, 036, 037, 038, 039, 045, 046, 047, 048, 049, 056, 057, 058, 059,
067, 068, 069, 078, 079, 089, 123, 124, 125, 126, 127, 128, 129, 134, 135,
136, 137, 138, 139, 145, 146, 147, 148, 149, 156, 157, 158, 159, 167, 168,
169, 178, 179, 189, 234, 235, 236, 237, 238, 239, 245, 246, 247, 248, 249,
256, 257, 258, 259, 267, 268, 269, 278, 279, 289, 345, 346, 347, 348, 349,
356, 357, 358, 359, 367, 368, 369, 378, 379, 389, 456, 457, 458, 459, 467,
468, 469, 478, 479, 489, 567, 568, 569, 578, 579, 589, 678, 679, 689, 789

julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `101-print_comb4.c`

Done! Help Check your code Get a sandbox QA Review

12. Software is eating the World

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints all possible combinations of two two-digit numbers.

- The numbers should range from `0` to `99`
- The two numbers should be separated by a space
- All numbers should be printed with two digits. `1` should be printed as `01`
- The combination of numbers must be separated by comma, followed by a space
- The combinations of numbers should be printed in ascending order
- `00 01` and `01 00` are considered as the same combination of the numbers `0` and `1`
- You can only use the `putchar` function (every other function (`printf`, `puts`, etc…) is forbidden)
- You can only use `putchar` eight times maximum in your code
- You are not allowed to use any variable of type `char`
- All your code should be in the `main` function

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 102-pr
int_comb5.c -o 102-print_comb5

julien@ubuntu:~/0x01$ ./102-print_comb5

00 01, 00 02, 00 03, 00 04, 00 05, 00 06, 00 07, 00 08, 00 09, 00 10, 00 11
, [...] 40 91, 40 92, 40 93, 40 94, 40 95, 40 96, 40 97, 40 98, 40 99, 41 4
2, 41 43, 41 44, 41 45, 41 46, 41 47, 41 48, 41 49, 41 50, 41 51, 41 52, 41
53 [...] 93 95, 93 96, 93 97, 93 98, 93 99, 94 95, 94 96, 94 97, 94 98, 94
99, 95 96, 95 97, 95 98, 95 99, 96 97, 96 98, 96 99, 97 98, 97 99, 98 99
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `102-print_comb5.c`

0x02. C - Functions, nested loops

C

- By: Julien Barbier
- Weight: 1
- Project over - took place from Dec 13, 2022 6:00 AM to Dec 14, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 106.0/106 mandatory & 36.0/36 optional
- **Altogether: 200.0%**
  - Mandatory: 100.0%
  - Optional: 100.0%
  - Calculation: 100.0% + (100.0% * 100.0%) == **200.0%**

Resources

Read or watch:

- Nested while loops
- C - Functions
- Learning to Program in C (Part 06) (*stop at 14:00*)
- What is the purpose of a function prototype?
- C - Header Files (*stop before the “Once-Only Headers” paragraph*)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are nested loops and how to use them
- What is a function and how do you use functions
- What is the difference between a declaration and a definition of a function
- What is a prototype
- Scope of variables
- What are the `gcc` flags `-Wall -Werror -pedantic -Wextra -std=gnu89`
- What are header files and how to to use them with `#include`

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`

- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, etc… is forbidden
- You are allowed to use _putchar
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don't forget to push your header file

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

More Info

You do not have to understand the call by reference (address), stack, static variables, recursions or arrays, yet.

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. _putchar

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints `_putchar`, followed by a new line.

- The program should return `0`

```
julien@ubuntu:~/0x02$  gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putc
har.c 0-putchar.c -o 0-putchar

julien@ubuntu:~/0x02$ ./0-putchar
```

```
_putchar
julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `0-putchar.c`

Done! Help Check your code Get a sandbox QA Review

1. I sometimes suffer from insomnia. And when I can't fall asleep, I play what I call the alphabet game

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the alphabet, in lowercase, followed by a new line.

- Prototype: `void print_alphabet(void);`
- You can only use `_putchar` twice in your code

```
julien@ubuntu:~/0x02$ cat 1-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_alphabet();
    return (0);
}
julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 1-main.c 1-alphabet.c -o 1-alphabet
julien@ubuntu:~/0x02$ ./1-alphabet
abcdefghijklmnopqrstuvwxyz
julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `1-alphabet.c`

Done! Help Check your code Get a sandbox QA Review

2. 10 x alphabet

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints 10 times the alphabet, in lowercase, followed by a new line.

- Prototype: `void print_alphabet_x10(void);`
- You can only use `_putchar` twice in your code

```
julien@ubuntu:~/0x02$ cat 2-main.c
#include "main.h"


/**
 * main - check the code.
 *
 * Return: Always 0.
 */
int main(void)
{
    print_alphabet_x10();
    return (0);
}
julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 2-main.c 2-print_alphabet_x10.c -o 2-alphabet_x10
julien@ubuntu:~/0x02$ ./2-alphabet_x10
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

```
abcdefghijklmnopqrstuvwxyz

abcdefghijklmnopqrstuvwxyz

julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `2-print_alphabet_x10.c`

Done! Help Check your code Get a sandbox QA Review

3. islower

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that checks for lowercase character.

- Prototype: `int _islower(int c);`
- Returns `1` if `c` is lowercase
- Returns `0` otherwise

FYI: The standard library provides a similar function: `islower`. Run `man islower` to learn more.

```
julien@ubuntu:~/0x02$ cat 3-main.c

#include "main.h"


/**

 * main - check the code.

 *

 * Return: Always 0.

 */

int main(void)

{

    int r;


    r = _islower('H');

    _putchar(r + '0');

    r = _islower('o');

    _putchar(r + '0');

    r = _islower(108);

    _putchar(r + '0');
    _putchar('\n');

    return (0);

}

julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 3-main.c 3-islower.c -o 3-islower

julien@ubuntu:~/0x02$ ./3-islower

011

julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `3-islower.c`

Done! Help Check your code Get a sandbox QA Review

4. isalpha

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that checks for alphabetic character.

- Prototype: `int _isalpha(int c);`
- Returns `1` if `c` is a letter, lowercase or uppercase
- Returns `0` otherwise

FYI: The standard library provides a similar function: `isalpha`. Run `man isalpha` to learn more.

```
julien@ubuntu:~/0x02$ cat 4-main.c

#include "main.h"


/**

 * main - check the code.

 *

 * Return: Always 0.

 */

int main(void)

{

    int r;


    r = _isalpha('H');
    _putchar(r + '0');
    r = _isalpha('o');
    _putchar(r + '0');
    r = _isalpha(108);
    _putchar(r + '0');
    r = _isalpha(';');
    _putchar(r + '0');
    _putchar('\n');
    return (0);
}
julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 4-main.c 4-isalpha.c -o 4-isalpha
julien@ubuntu:~/0x02$ ./4-isalpha
1110
julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `4-isalpha.c`

Done! Help Check your code Get a sandbox QA Review

5. Sign

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the sign of a number.

- Prototype: `int print_sign(int n);`
- Returns `1` and prints `+` if `n` is greater than zero
- Returns `0` and prints `0` if `n` is zero
- Returns `-1` and prints `-` if `n` is less than zero

```
julien@ubuntu:~/0x02$ cat 5-main.c
#include "main.h"

/**
 * main - check the code.
 *
 * Return: Always 0.
 */
int main(void)
{
    int r;

    r = print_sign(98);
    _putchar(',');
    _putchar(' ');
    _putchar(r + '0');
    _putchar('\n');
    r = print_sign(0);
    _putchar(',');
    _putchar(' ');
    _putchar(r + '0');
    _putchar('\n');
    r = print_sign(0xff);
    _putchar(',');
    _putchar(' ');
    _putchar(r + '0');
    _putchar('\n');
    r = print_sign(-1);
    _putchar(',');
    _putchar(' ');
    _putchar(r + '0');
    _putchar('\n');
    return (0);
}
julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 5-main.c 5-sign.c -o 5-sign
julien@ubuntu:~/0x02$ ./5-sign
+, 1
0, 0
+, 1
```

```
-, /
julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `5-sign.c`

Done! Help Check your code Get a sandbox QA Review

6. There is no such thing as absolute value in this world. You can only estimate what a thing is worth to you

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that computes the absolute value of an integer.

- Prototype: `int _abs(int);`

FYI: The standard library provides a similar function: `abs`. Run `man abs` to learn more.

```
julien@ubuntu:~/0x02$ cat 6-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int r;

    r = _abs(-1);
    printf("%d\n", r);
    r = _abs(0);
    printf("%d\n", r);
    r = _abs(1);
    printf("%d\n", r);
    r = _abs(-98);
    printf("%d\n", r);
    return (0);
}
julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 6-main.c 6-abs.c -o 6-abs
julien@ubuntu:~/0x02$ ./6-abs
1
0
1
98
julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `6-abs.c`

Done! Help Check your code Get a sandbox QA Review

7. There are only 3 colors, 10 digits, and 7 notes; it's what we do with them that's important

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the last digit of a number.

- Prototype: `int print_last_digit(int);`
- Returns the value of the last digit

```
julien@ubuntu:~/0x02$ cat 7-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int r;


    print_last_digit(98);
    print_last_digit(0);
    r = print_last_digit(-1024);
    _putchar('0' + r);
    _putchar('\n');
    return (0);
}
julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 7-main.c 7-print_last_digit.c -o 7-last_digit
julien@ubuntu:~/0x02$ ./7-last_digit
8044
julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `7-print_last_digit.c`

Done! Help Check your code Get a sandbox QA Review

8. I'm federal agent Jack Bauer, and today is the longest day of my life

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints every minute of the day of Jack Bauer, starting from 00:00 to 23:59.

- Prototype: `void jack_bauer(void);`
- You can listen to this soundtrack while coding :)

```
julien@ubuntu:~/0x02$ cat 8-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
```

```
int main(void)

{

    jack_bauer();

    return (0);

}

julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 8-main.c 8-24_hours.c -o 8-24

julien@ubuntu:~/0x02$ ./8-24 | head

00:00

00:01

00:02

00:03

00:04

00:05

00:06

00:07

00:08

00:09

julien@ubuntu:~/0x02$ ./8-24 | tail

23:50

23:51

23:52

23:53

23:54

23:55

23:56

23:57

23:58

23:59

julien@ubuntu:~/0x02$ ./8-24 | wc -l

1440

julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `8-24_hours.c`

Done! Help Check your code Get a sandbox QA Review

9. Learn your times table

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the 9 times table, starting with 0.

- Prototype: `void times_table(void);`
- Format: see example

```
julien@ubuntu:~/0x02$ cat 9-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    times_table();
    return (0);
}
julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 9-main.c 9-times_table.c -o 9-times_table
ulien@ubuntu:~/0x02$ ./9-times_table | cat -e
0,  0,  0,  0,  0,  0,  0,  0,  0,  0$
0,  1,  2,  3,  4,  5,  6,  7,  8,  9$
0,  2,  4,  6,  8, 10, 12, 14, 16, 18$
0,  3,  6,  9, 12, 15, 18, 21, 24, 27$
0,  4,  8, 12, 16, 20, 24, 28, 32, 36$
0,  5, 10, 15, 20, 25, 30, 35, 40, 45$
0,  6, 12, 18, 24, 30, 36, 42, 48, 54$
0,  7, 14, 21, 28, 35, 42, 49, 56, 63$
```

```
0,  8, 16, 24, 32, 40, 48, 56, 64, 72$
0,  9, 18, 27, 36, 45, 54, 63, 72, 81$
julien@ubuntu:~/0x02$ ./9-times_table | tr ' ' . | cat -e
0,..0,..0,..0,..0,..0,..0,..0,..0,..0$
0,..1,..2,..3,..4,..5,..6,..7,..8,..9$
0,..2,..4,..6,..8,.10,.12,.14,.16,.18$
0,..3,..6,..9,.12,.15,.18,.21,.24,.27$
0,..4,..8,.12,.16,.20,.24,.28,.32,.36$
0,..5,.10,.15,.20,.25,.30,.35,.40,.45$
0,..6,.12,.18,.24,.30,.36,.42,.48,.54$
0,..7,.14,.21,.28,.35,.42,.49,.56,.63$
0,..8,.16,.24,.32,.40,.48,.56,.64,.72$
0,..9,.18,.27,.36,.45,.54,.63,.72,.81$
julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `9-times_table.c`

Done! Help Check your code Get a sandbox QA Review

10. a + b

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that adds two integers and returns the result.

- Prototype: `int add(int, int);`

```
julien@ubuntu:~/$ cat 10-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
```

```
int main(void)

{

    int n;


    n = add(89, 9);

    printf("%d\n", n);

    return (0);

}

julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 10-main.c 10-add.c -o 10-add

julien@ubuntu:~/0x02$ ./10-add

98

julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `10-add.c`

Done! Help Check your code Get a sandbox QA Review

11. 98 Battery Street, the OG

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints all natural numbers from `n` to `98`, followed by a new line.

- Prototype: `void print_to_98(int n);`
- Numbers must be separated by a comma, followed by a space
- Numbers should be printed in order
- The first printed number should be the number passed to your function
- The last printed number should be `98`
- You are allowed to use the standard library

```
julien@ubuntu:~/0x02$ cat 11-main.c

#include "main.h"


/**

 * main - check the code

 *

 * Return: Always 0.
 */
int main(void)
{
    print_to_98(0);
    print_to_98(98);
    print_to_98(111);
    print_to_98(81);
    print_to_98(-10);
    return (0);
}
julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 11-main.c 11-print_to_98.c -o 11-98
julien@ubuntu:~/0x02$ ./11-98
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
1, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96,
97, 98
98
111, 110, 109, 108, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98
-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86,
87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98
julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `11-print_to_98.c`

Done! Help Check your code Get a sandbox QA Review

12. The World looks like a multiplication-table, or a mathematical equation, which, turn it how you will, balances itself

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the `n` times table, starting with 0.

- Prototype: `void print_times_table(int n);`
- If `n` is greater than `15` or less than `0` the function should not print anything
- Format: see example

```
julien@ubuntu:~/0x02$ cat 100-main.c
#include "main.h"

/**
 * main - check the code.
 *
 * Return: Always 0.
 */
int main(void)
{
    print_times_table(3);
    _putchar('\n');
    print_times_table(5);
    _putchar('\n');
    print_times_table(98);
    _putchar('\n');
    print_times_table(12);
    return (0);
}
julien@ubuntu:~/0x02$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 100-main.c 100-times_table.c -o 100-times_table
julien@ubuntu:~/0x02$ ./100-times_table
0,   0,   0,   0
0,   1,   2,   3
0,   2,   4,   6
0,   3,   6,   9

0,   0,   0,   0,   0,   0
0,   1,   2,   3,   4,   5
0,   2,   4,   6,   8,  10
```

```
0,   3,   6,   9,  12,  15
0,   4,   8,  12,  16,  20
0,   5,  10,  15,  20,  25


0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0
0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12
0,   2,   4,   6,   8,  10,  12,  14,  16,  18,  20,  22,  24
0,   3,   6,   9,  12,  15,  18,  21,  24,  27,  30,  33,  36
0,   4,   8,  12,  16,  20,  24,  28,  32,  36,  40,  44,  48
0,   5,  10,  15,  20,  25,  30,  35,  40,  45,  50,  55,  60
0,   6,  12,  18,  24,  30,  36,  42,  48,  54,  60,  66,  72
0,   7,  14,  21,  28,  35,  42,  49,  56,  63,  70,  77,  84
0,   8,  16,  24,  32,  40,  48,  56,  64,  72,  80,  88,  96
0,   9,  18,  27,  36,  45,  54,  63,  72,  81,  90,  99, 108
0,  10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110, 120
0,  11,  22,  33,  44,  55,  66,  77,  88,  99, 110, 121, 132
0,  12,  24,  36,  48,  60,  72,  84,  96, 108, 120, 132, 144
julien@ubuntu:~/0x02$ ./100-times_table | tr ' ' . | cat -e
0,...0,...0,...0$
0,...1,...2,...3$
0,...2,...4,...6$
0,...3,...6,...9$
$
0,...0,...0,...0,...0,...0$
0,...1,...2,...3,...4,...5$
0,...2,...4,...6,...8,..10$
0,...3,...6,...9,..12,..15$
0,...4,...8,..12,..16,..20$
0,...5,..10,..15,..20,..25$
$
$
0,...0,...0,...0,...0,...0,...0,...0,...0,...0,...0,...0,...0$
```

```
0,...1,...2,...3,...4,...5,...6,...7,...8,...9,..10,..11,..12$
0,...2,...4,...6,...8,..10,..12,..14,..16,..18,..20,..22,..24$
0,...3,...6,...9,..12,..15,..18,..21,..24,..27,..30,..33,..36$
0,...4,...8,..12,..16,..20,..24,..28,..32,..36,..40,..44,..48$
0,...5,..10,..15,..20,..25,..30,..35,..40,..45,..50,..55,..60$
0,...6,..12,..18,..24,..30,..36,..42,..48,..54,..60,..66,..72$
0,...7,..14,..21,..28,..35,..42,..49,..56,..63,..70,..77,..84$
0,...8,..16,..24,..32,..40,..48,..56,..64,..72,..80,..88,..96$
0,...9,..18,..27,..36,..45,..54,..63,..72,..81,..90,..99,.108$
0,..10,..20,..30,..40,..50,..60,..70,..80,..90,.100,.110,.120$
0,..11,..22,..33,..44,..55,..66,..77,..88,..99,.110,.121,.132$
0,..12,..24,..36,..48,..60,..72,..84,..96,.108,.120,.132,.144$
julien@ubuntu:~/0x02$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `100-times_table.c`

Done! Help Check your code Get a sandbox QA Review

13. Nature made the natural numbers; All else is the work of women

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

If we list all the natural numbers below `10` that are multiples of `3` or `5`, we get `3`, `5`, `6` and `9`. The sum of these multiples is `23`. Write a program that computes and prints the sum of all the multiples of `3` or `5` below `1024` (excluded), followed by a new line.

- You are allowed to use the standard library

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `101-natural.c`

Done! Help Check your code Get a sandbox QA Review

14. In computer class, the first assignment was to write a program to print the first 100 Fibonacci numbers. Instead, I wrote a program that would steal passwords of students. My teacher gave me an A

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints the first 50 Fibonacci numbers, starting with `1` and `2`, followed by a new line.

- The numbers must be separated by comma, followed by a space `, `
- You are allowed to use the standard library

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `102-fibonacci.c`

Done! Help Check your code Get a sandbox QA Review

15. Even Liber Abbaci

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with `1` and `2`, the first 10 terms will be: `1, 2, 3, 5, 8, 13, 21, 34, 55, 89`. By considering the terms in the Fibonacci sequence whose values do not exceed 4,000,000, write a program that finds and prints the sum of the even-valued terms, followed by a new line.

- You are allowed to use the standard library

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `103-fibonacci.c`

Done! Help Check your code Get a sandbox QA Review

16. In computer class, the first assignment was to write a program to print the first 100 Fibonacci numbers. Instead, I wrote a program that would steal passwords of students. My teacher gave me an A+

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that finds and prints the first 98 Fibonacci numbers, starting with `1` and `2`, followed by a new line.

- The numbers should be separated by comma, followed by a space `, `
- You are allowed to use the standard library
- You are not allowed to use any other library (You can't use `GMP` etc…)
- You are not allowed to use `long long`, `malloc`, pointers, arrays/tables, or structures
- You are not allowed to hard code any Fibonacci number (except for `1` and `2`)

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x02-functions_nested_loops`
- File: `104-fibonacci.c`

0x03. C - Debugging

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, without the help of Google:

General

- What is debugging
- What are some methods of debugging manually
- How to read the error messages

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- Your code should use the `Betty` style. It will be checked using `betty-style.pl` and `betty-doc.pl`
- A README.md file at the root of the repo, containing a description of the repository
- A README.md file, at the root of the folder of this project (i.e. `0x03-debugging`), describing what this project is about

Quiz questions
Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Multiple mains

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In most projects, we often give you only one main file to test with. For example, this main file is a test for a `postitive_or_negative()` function similar to the one you worked with in an earlier C project:

```
carrie@ubuntu:/debugging$ cat main.c
#include "main.h"

/**
* main - tests function that prints if integer is positive or negative
* Return: 0
*/

int main(void)
{
        int i;

        i = 98;
        positive_or_negative(i);

        return (0);
}
carrie@ubuntu:/debugging$
carrie@ubuntu:/debugging$ cat main.h
#ifndef MAIN_H
#define MAIN_H

#include <stdio.h>

void positive_or_negative(int i);

#endif /* MAIN_H */
carrie@ubuntu:/debugging$
carrie@ubuntu:/debugging$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 po
sitive_or_negative.c main.c
carrie@ubuntu:/debugging$ ./a.out
```

```
98 is positive

carrie@ubuntu:/debugging$
```

Based on the `main.c` file above, create a file named `0-main.c`. This file must test that the function `positive_or_negative()` gives the correct output when given a case of `0`.

You are not coding the solution / function, you're just testing it! However, you can adapt your function from 0x01. C - Variables, if, else, while - Task #0 to compile with this main file to test locally.

- You only need to upload `0-main.c` and `main.h` for this task. We will provide our own `positive_or_negative()` function.
- You are not allowed to add or remove lines of code, you may change only **one** line in this task.

```
carrie@ubuntu:/debugging$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 po
sitive_or_negative.c 0-main.c -o 0-main

carrie@ubuntu:/debugging$ ./0-main

0 is zero

carrie@ubuntu:/debugging$ wc -l 0-main.c

16 1-main.c

carrie@ubuntu:/debugging$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x03-debugging`
- File: `0-main.c, main.h`

Done! Help Check your code QA Review

1. Like, comment, subscribe

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Copy this main file. Comment out (don't delete it!) the part of the code that is causing the output to go into an infinite loop.

- Don't add or remove any lines of code, as we will be checking your line count. You are only allowed to comment out existing code.
- You do not have to compile with `-Wall -Werror -Wextra -pedantic` for this task.

```
carrie@ubuntu:/debugging$ cat 1-main.c

#include <stdio.h>


/**

* main - causes an infinite loop
* Return: 0
*/

int main(void)
{
        int i;

        printf("Infinite loop incoming :(\n");

        i = 0;

        while (i < 10)
        {
                putchar(i);
        }

        printf("Infinite loop avoided! \\o/\n");

        return (0);
}
carrie@ubuntu:/debugging$
```

Your output should look like this:

```
carrie@ubuntu:/debugging$ gcc -std=gnu89 1-main.c -o 1-main
carrie@ubuntu:/debugging$ ./1-main
Infinite loop incoming :(
Infinite loop avoided! \o/
carrie@ubuntu:/debugging$ wc -l 1-main.c
24 1-main.c
carrie@ubuntu:/debugging$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x03-debugging`
- File: `1-main.c`

Done! Help Check your code Get a sandbox QA Review

2. 0 > 972?

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

This program prints the largest of three integers.

```
carrie@ubuntu:/debugging$ cat 2-main.c
#include <stdio.h>
#include "main.h"

/**
* main - prints the largest of 3 integers
* Return: 0
*/

int main(void)
{
        int a, b, c;
        int largest;

        a = 972;
        b = -98;
        c = 0;

        largest = largest_number(a, b, c);

        printf("%d is the largest number\n", largest);

        return (0);
}
carrie@ubuntu:/debugging$
carrie@ubuntu:/debugging$ cat 2-largest_number.c
#include "main.h"
```

```
/**
 * largest_number - returns the largest of 3 numbers
 * @a: first integer
 * @b: second integer
 * @c: third integer
 * Return: largest number
 */

int largest_number(int a, int b, int c)
{
    int largest;

    if (a > b && b > c)
    {
        largest = a;
    }
    else if (b > a && a > c)
    {
        largest = b;
    }
    else
    {
        largest = c;
    }

    return (largest);
}

carrie@ubuntu:/debugging$
carrie@ubuntu:/debugging$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 2-
largest_number.c 2-main.c -o 2-main
carrie@ubuntu:/debugging$ ./2-main
0 is the largest number
```

```
carrie@ubuntu:/debugging$
```

? That’s definitely not right.

Fix the code in `2-largest_number.c` so that it correctly prints out the largest of three numbers, no matter the case.

- Line count will not be checked for this task.

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x03-debugging`
- File: `2-largest_number.c, main.h`

Done! Help Check your code Get a sandbox QA Review

3. Leap year

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

This program converts a date to the day of year and determines how many days are left in the year, taking leap year into consideration.

```
carrie@ubuntu:/debugging$ cat 3-main_a.c
#include <stdio.h>
#include "main.h"

/**
* main - takes a date and prints how many days are left in the year, taking
* leap years into account
* Return: 0
*/

int main(void)
{
    int month;
    int day;
    int year;

    month = 4;
    day = 01;
    year = 1997;
    printf("Date: %02d/%02d/%04d\n", month, day, year);

    day = convert_day(month, day);

    print_remaining_days(month, day, year);

    return (0);
}

carrie@ubuntu:/debugging$
carrie@ubuntu:/debugging$ cat 3-convert_day.c
#include "main.h"

/**
* convert_day - converts day of month to day of year, without accounting
* for leap year
* @month: month in number format
* @day: day of month
* Return: day of year
*/

int convert_day(int month, int day)
{
    switch (month)
    {
        case 2:
            day = 31 + day;
            break;
        case 3:
            day = 59 + day;
            break;
        case 4:
            day = 90 + day;
            break;
        case 5:
            day = 120 + day;
            break;
        case 6:
            day = 151 + day;
            break;
        case 7:
            day = 181 + day;
            break;
        case 8:
            day = 212 + day;
            break;
        case 9:
            day = 243 + day;
            break;
        case 10:
            day = 273 + day;
            break;
        case 11:
            day = 304 + day;
            break;
        case 12:
            day = 334 + day;
            break;
        default:
            break;
    }
    return (day);
}

carrie@ubuntu:/debugging$
```

```
carrie@ubuntu:/debugging$ cat 3-print_remaining_days.c
#include <stdio.h>
#include "main.h"

/**
* print_remaining_days - takes a date and prints how many days are
* left in the year, taking leap years into account
* @month: month in number format
* @day: day of month
* @year: year
* Return: void
*/

void print_remaining_days(int month, int day, int year)
{
    if ((year % 4 == 0 || year % 400 == 0) && !(year % 100 == 0))
    {
        if (month >= 2 && day >= 60)
        {
            day++;
        }

        printf("Day of the year: %d\n", day);
        printf("Remaining days: %d\n", 366 - day);
    }
    else
    {
        if (month == 2 && day == 60)
        {
            printf("Invalid date: %02d/%02d/%04d\n", month, day - 31, year)
;
        }
        else
        {
            printf("Day of the year: %d\n", day);

            printf("Remaining days: %d\n", 365 - day);

        }

    }

}


carrie@ubuntu:/debugging$

carrie@ubuntu:/debugging$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 3-
convert_day.c 3-print_remaining_days.c 3-main_a.c -o 3-main_a

carrie@ubuntu:/debugging$ ./3-main_a

Date: 04/01/1997

Day of the year: 91

Remaining days: 274

carrie@ubuntu:/debugging$
```

Output looks good for `04/01/1997`! Let's make a new main file `3-main_b.c` to try a case that is a leap year: `02/29/2000`.

```
carrie@ubuntu:/debugging$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 3-
convert_day.c 3-print_remaining_days.c 3-main_b.c -o 3-main_b

carrie@ubuntu:/debugging$ ./3-main_b

Date: 02/29/2000

Invalid date: 02/29/2000

carrie@ubuntu:/debugging$
```

? That doesn't seem right.

Fix the `print_remaining_days()` function so that the output works correctly for *all* dates and *all* leap years.

- Line count will not be checked for this task.
- You can assume that all test cases have valid months (i.e. the value of `month` will never be less than `1` or greater than `12`) and valid days (i.e. the value of `day` will never be less than `1` or greater than `31`).
- You can assume that all test cases have valid month/day combinations (i.e. there will never be a June 31st or November 31st, etc.), but not all month/day/year combinations are valid (i.e. February 29, 1991 or February 29, 2427).

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x03-debugging`
- File: `3-print_remaining_days.c, main.h`

0x04. C - More functions, more nested loops

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, etc… is forbidden
- You are allowed to use _putchar
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don't forget to push your header file

More Info

You do not have to understand the call by reference (address), stack, static variables, recursions or arrays, yet.

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. isupper

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that checks for uppercase character.

- Prototype: `int _isupper(int c);`
- Returns `1` if `c` is uppercase

- Returns `0` otherwise

FYI: The standard library provides a similar function: `isupper`. Run `man isupper` to learn more.

```
julien@ubuntu:~/0x04$ cat 0-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code.
 *
 * Return: Always 0.
 */
int main(void)
{
    char c;

    c = 'A';
    printf("%c: %d\n", c, _isupper(c));
    c = 'a';
    printf("%c: %d\n", c, _isupper(c));
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-main
.c 0-isupper.c -o 0-isuper
julien@ubuntu:~/0x04$ ./0-isuper
A: 1
a: 0
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `0-isupper.c`

Done! Help Check your code Get a sandbox QA Review

1. isdigit

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that checks for a digit (`0` through `9`).

- Prototype: `int _isdigit(int c);`
- Returns `1` if `c` is a digit
- Returns `0` otherwise

FYI: The standard library provides a similar function: isdigit. Run man isdigit to learn more.

```
julien@ubuntu:~/0x04$ cat 1-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char c;

    c = '0';
    printf("%c: %d\n", c, _isdigit(c));
    c = 'a';
    printf("%c: %d\n", c, _isdigit(c));
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-main.c 1-isdigit.c -o 1-isdigit
julien@ubuntu:~/0x04$ ./1-isdigit
0: 1
a: 0
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`

- Directory: `0x04-more_functions_nested_loops`
- File: `1-isdigit.c`

Done! Help Check your code Get a sandbox QA Review

2. Collaboration is multiplication

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that multiplies two integers.

- Prototype: `int mul(int a, int b);`

```
julien@ubuntu:~/0x04$ cat 2-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    printf("%d\n", mul(98, 1024));
    printf("%d\n", mul(-402, 4096));
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-main
.c 2-mul.c -o 2-mul
julien@ubuntu:~/0x04$ ./2-mul
100352
-1646592
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `2-mul.c`

Done! Help Check your code Get a sandbox QA Review

3. The numbers speak for themselves

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the numbers, from `0` to `9`, followed by a new line.

- Prototype: `void print_numbers(void);`
- You can only use `_putchar` twice in your code

```
julien@ubuntu:~/0x04$ cat 3-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_numbers();
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 3-main.c 3-print_numbers.c -o 3-print_numbers
julien@ubuntu:~/0x04$ ./3-print_numbers | cat -e
0123456789$
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `3-print_numbers.c`

Done! Help Check your code Get a sandbox QA Review

4. I believe in numbers and signs

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the numbers, from `0` to `9`, followed by a new line.

- Prototype: `void print_most_numbers(void);`
- Do not print `2` and `4`
- You can only use `_putchar` twice in your code

```
julien@ubuntu:~/0x04$ cat 4-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_most_numbers();
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 4-main.c 4-print_most_numbers.c -o 4-print_most_numbers
julien@ubuntu:~/0x04$ ./4-print_most_numbers
01356789
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `4-print_most_numbers.c`

Done! Help Check your code Get a sandbox QA Review

5. Numbers constitute the only universal language

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints 10 times the numbers, from `0` to `14`, followed by a new line.

- Prototype: `void more_numbers(void);`
- You can only use `_putchar` three times in your code

```
julien@ubuntu:~/0x04$ cat 5-main.c
#include "main.h"
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    more_numbers();
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 5-main.c 5-more_numbers.c -o 5-more_numbers
julien@ubuntu:~/0x04$ ./5-more_numbers
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
julien@ubuntu:~/0x04
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `5-more_numbers.c`

Done! Help Check your code Get a sandbox QA Review

6. The shortest distance between two points is a straight line

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that draws a straight line in the terminal.

- Prototype: `void print_line(int n);`
- You can only use `_putchar` function to print
- Where `n` is the number of times the character `_` should be printed
- The line should end with a `\n`
- If `n` is `0` or less, the function should only print `\n`

```
julien@ubuntu:~/0x04$ cat 6-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_line(0);
    print_line(2);
    print_line(10);
    print_line(-4);
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 6-main.c 6-print_line.c -o 6-lines
julien@ubuntu:~/0x04$ ./6-lines | cat -e
$
__$
__________$
$
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `6-print_line.c`

Done! Help Check your code Get a sandbox QA Review

7. I feel like I am diagonally parked in a parallel universe

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that draws a diagonal line on the terminal.

- Prototype: `void print_diagonal(int n);`
- You can only use `_putchar` function to print
- Where `n` is the number of times the character `\` should be printed
- The diagonal should end with a `\n`
- If `n` is `0` or less, the function should only print a `\n`

```
julien@ubuntu:~/0x04$ cat 7-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_diagonal(0);
    print_diagonal(2);
    print_diagonal(10);
    print_diagonal(-4);
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 7-main.c 7-print_diagonal.c -o 7-diagonals
julien@ubuntu:~/0x04$ ./7-diagonals | cat -e
$
\$
 \$
\$
 \$
  \$
   \$
```

```
    \$
     \$
      \$
       \$
        \$
         \$
$
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `7-print_diagonal.c`

Done! Help Check your code Get a sandbox QA Review

8. You are so much sunshine in every square inch

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a square, followed by a new line.

- Prototype: `void print_square(int size);`
- You can only use `_putchar` function to print
- Where `size` is the size of the square
- If `size` is `0` or less, the function should print only a new line
- Use the character `#` to print the square

```
julien@ubuntu:~/0x04$ cat 8-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_square(2);
    print_square(10);
    print_square(0);
    return (0);
}

julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 8-main.c 8-print_square.c -o 8-squares
julien@ubuntu:~/0x04$ ./8-squares
##
##
##########
##########
##########
##########
##########
##########
##########
##########
##########
##########

julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `8-print_square.c`

Done! Help Check your code Get a sandbox QA Review

9. Fizz-Buzz

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

The “Fizz-Buzz test” is an interview question designed to help filter out the 99.5% of programming job candidates who can’t seem to program their way out of a wet paper bag.

Write a program that prints the numbers from `1` to `100`, followed by a new line. But for multiples of three print `Fizz` instead of the number and for the multiples of five print `Buzz`. For numbers which are multiples of both three and five print `FizzBuzz`.

- Each number or word should be separated by a space

- You are allowed to use the standard library

```
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 9-fizz
_buzz.c -o 9-fizz_buzz

julien@ubuntu:~/0x04$ ./9-fizz_buzz

1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19 Buz
z Fizz 22 23 Fizz Buzz 26 Fizz 28 29 FizzBuzz 31 32 Fizz 34 Buzz Fizz 37 38
Fizz Buzz 41 Fizz 43 44 FizzBuzz 46 47 Fizz 49 Buzz Fizz 52 53 Fizz Buzz 56
Fizz 58 59 FizzBuzz 61 62 Fizz 64 Buzz Fizz 67 68 Fizz Buzz 71 Fizz 73 74 F
izzBuzz 76 77 Fizz 79 Buzz Fizz 82 83 Fizz Buzz 86 Fizz 88 89 FizzBuzz 91 9
2 Fizz 94 Buzz Fizz 97 98 Fizz Buzz

julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `9-fizz_buzz.c`

Done! Help Check your code Get a sandbox QA Review

10. Triangles

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a triangle, followed by a new line.

- Prototype: `void print_triangle(int size);`
- You can only use `_putchar` function to print
- Where `size` is the size of the triangle
- If `size` is `0` or less, the function should print only a new line
- Use the character `#` to print the triangle

```
julien@ubuntu:~/0x04$ cat 10-main.c

#include "main.h"


/**

 * main - check the code

 *

 * Return: Always 0.

 */

int main(void)

{

    print_triangle(2);
    print_triangle(10);
    print_triangle(1);
    print_triangle(0);
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 10-main.c 10-print_triangle.c -o 10-triangles
julien@ubuntu:~/0x04$ ./10-triangles
 #
##
         #
        ##
       ###
      ####
     #####
    ######
   #######
  ########
 #########
##########
#

julien@ubuntu:~/0x04$ ./10-triangles | tr ' ' . | cat -e
.#$
##$
.........#$
........##$
.......###$
......####$
.....#####$
....######$
...#######$
..########$
.#########$
```

```
##########$

#$

$

julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `10-print_triangle.c`

Done! Help Check your code Get a sandbox QA Review

11. The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

The prime factors of `1231952` are `2`, `2`, `2`, `2`, `37` and `2081`.

Write a program that finds and prints the largest prime factor of the number `612852475143`, followed by a new line.

- You are allowed to use the standard library
- Your program will be compiled with this command: `gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-prime_factor.c -o 100-prime_factor -lm`

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `100-prime_factor.c`

Done! Help Check your code Get a sandbox QA Review

12. Numbers have life; they're not just symbols on paper

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints an integer.

- Prototype: `void print_number(int n);`
- You can only use `_putchar` function to print
- You are not allowed to use `long`
- You are not allowed to use arrays or pointers
- You are not allowed to hard-code special values

```
julien@ubuntu:~/0x04$ cat 101-main.c

#include "main.h"
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_number(98);
    _putchar('\n');
    print_number(402);
    _putchar('\n');
    print_number(1024);
    _putchar('\n');
    print_number(0);
    _putchar('\n');
    print_number(-98);
    _putchar('\n');
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 101-main.c 101-print_number.c -o 101-print_numbers
julien@ubuntu:~/0x04$ ./101-print_numbers
98
402
1024
0
-98
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `101-print_number.c`

0x04. C - More functions, more nested loops

Resources

Read or watch:

- Nested while loops
- C - Functions
- Learning to Program in C (Part 06) (*stop at 14:00*)
- What is the purpose of a function prototype?
- C - Header Files (*stop before the "Once-Only Headers" paragraph*)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are nested loops and how to use them
- What is a function and how do you use functions
- What is the difference between a declaration and a definition of a function
- What is a prototype
- Scope of variables
- What are the `gcc` flags `-Wall -Werror -pedantic -Wextra -std=gnu89`
- What are header files and how to to use them with `#include`

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory

- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, etc… is forbidden
- You are allowed to use _putchar
- You don’t have to push `_putchar.c`, we will use our file. If you do it won’t be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don’t have to push them to your repo (if you do we won’t take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don’t forget to push your header file

More Info

You do not have to understand the call by reference (address), stack, static variables, recursions or arrays, yet.

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. isupper

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that checks for uppercase character.

- Prototype: `int _isupper(int c);`
- Returns `1` if `c` is uppercase
- Returns `0` otherwise

FYI: The standard library provides a similar function: `isupper`. Run `man isupper` to learn more.

```
julien@ubuntu:~/0x04$ cat 0-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code.
 *
 * Return: Always 0.
 */
int main(void)
{
    char c;

    c = 'A';
    printf("%c: %d\n", c, _isupper(c));
    c = 'a';
    printf("%c: %d\n", c, _isupper(c));
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-main
.c 0-isupper.c -o 0-isuper
julien@ubuntu:~/0x04$ ./0-isuper
A: 1
a: 0
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `0-isupper.c`

Done! Help Check your code Get a sandbox QA Review

1. isdigit

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that checks for a digit (`0` through `9`).

- Prototype: `int _isdigit(int c);`
- Returns `1` if `c` is a digit
- Returns `0` otherwise

FYI: The standard library provides a similar function: isdigit. Run man isdigit to learn more.

```
julien@ubuntu:~/0x04$ cat 1-main.c
#include "main.h"
#include <stdio.h>

```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char c;

    c = '0';
    printf("%c: %d\n", c, _isdigit(c));
    c = 'a';
    printf("%c: %d\n", c, _isdigit(c));
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-main
.c 1-isdigit.c -o 1-isdigit
julien@ubuntu:~/0x04$ ./1-isdigit
0: 1
a: 0
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `1-isdigit.c`

Done! Help Check your code Get a sandbox QA Review

2. Collaboration is multiplication

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that multiplies two integers.

- Prototype: `int mul(int a, int b);`

```
julien@ubuntu:~/0x04$ cat 2-main.c
#include "main.h"
```

```
#include <stdio.h>


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    printf("%d\n", mul(98, 1024));
    printf("%d\n", mul(-402, 4096));
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-main
.c 2-mul.c -o 2-mul
julien@ubuntu:~/0x04$ ./2-mul
100352
-1646592
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `2-mul.c`

Done! Help Check your code Get a sandbox QA Review

3. The numbers speak for themselves

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the numbers, from `0` to `9`, followed by a new line.

- Prototype: `void print_numbers(void);`
- You can only use `_putchar` twice in your code

```
julien@ubuntu:~/0x04$ cat 3-main.c
#include "main.h"
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_numbers();
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 3-main.c 3-print_numbers.c -o 3-print_numbers
julien@ubuntu:~/0x04$ ./3-print_numbers | cat -e
0123456789$
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `3-print_numbers.c`

Done! Help Check your code Get a sandbox QA Review

4. I believe in numbers and signs

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the numbers, from `0` to `9`, followed by a new line.

- Prototype: `void print_most_numbers(void);`
- Do not print `2` and `4`
- You can only use `_putchar` twice in your code

```
julien@ubuntu:~/0x04$ cat 4-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_most_numbers();
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 4-main.c 4-print_most_numbers.c -o 4-print_most_numbers
julien@ubuntu:~/0x04$ ./4-print_most_numbers
01356789
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `4-print_most_numbers.c`

Done! Help Check your code Get a sandbox QA Review

5. Numbers constitute the only universal language

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints 10 times the numbers, from `0` to `14`, followed by a new line.

- Prototype: `void more_numbers(void);`
- You can only use `_putchar` three times in your code

```
julien@ubuntu:~/0x04$ cat 5-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    more_numbers();
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 5-main.c 5-more_numbers.c -o 5-more_numbers
julien@ubuntu:~/0x04$ ./5-more_numbers
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
01234567891011121314
julien@ubuntu:~/0x04
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `5-more_numbers.c`

Done! Help Check your code Get a sandbox QA Review

6. The shortest distance between two points is a straight line

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that draws a straight line in the terminal.

- Prototype: `void print_line(int n);`
- You can only use `_putchar` function to print
- Where `n` is the number of times the character `_` should be printed
- The line should end with a `\n`
- If `n` is `0` or less, the function should only print `\n`

```
julien@ubuntu:~/0x04$ cat 6-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_line(0);
    print_line(2);
    print_line(10);
    print_line(-4);
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 6-main.c 6-print_line.c -o 6-lines
julien@ubuntu:~/0x04$ ./6-lines | cat -e
$
__$
__________$
$
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `6-print_line.c`

Done! Help Check your code Get a sandbox QA Review

7. I feel like I am diagonally parked in a parallel universe

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that draws a diagonal line on the terminal.

- Prototype: `void print_diagonal(int n);`
- You can only use `_putchar` function to print
- Where `n` is the number of times the character `\` should be printed
- The diagonal should end with a `\n`
- If `n` is `0` or less, the function should only print a `\n`

```
julien@ubuntu:~/0x04$ cat 7-main.c
#include "main.h"
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_diagonal(0);
    print_diagonal(2);
    print_diagonal(10);
    print_diagonal(-4);
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 7-main.c 7-print_diagonal.c -o 7-diagonals
julien@ubuntu:~/0x04$ ./7-diagonals | cat -e
$
\$
 \$
\$
 \$
  \$
   \$
    \$
     \$
      \$
       \$
        \$
         \$
$
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `7-print_diagonal.c`

Done! Help Check your code Get a sandbox QA Review

8. You are so much sunshine in every square inch

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a square, followed by a new line.

- Prototype: `void print_square(int size);`
- You can only use `_putchar` function to print
- Where `size` is the size of the square
- If `size` is `0` or less, the function should print only a new line
- Use the character `#` to print the square

```
julien@ubuntu:~/0x04$ cat 8-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_square(2);
    print_square(10);
    print_square(0);
    return (0);
}

julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 8-main.c 8-print_square.c -o 8-squares
julien@ubuntu:~/0x04$ ./8-squares
##
##
##########
```

```
##########
##########
##########
##########
##########
##########
##########
##########
##########

julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `8-print_square.c`

Done! Help Check your code Get a sandbox QA Review

9. Fizz-Buzz

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

The “Fizz-Buzz test” is an interview question designed to help filter out the 99.5% of programming job candidates who can’t seem to program their way out of a wet paper bag.

Write a program that prints the numbers from `1` to `100`, followed by a new line. But for multiples of three print `Fizz` instead of the number and for the multiples of five print `Buzz`. For numbers which are multiples of both three and five print `FizzBuzz`.

- Each number or word should be separated by a space
- You are allowed to use the standard library

```
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 9-fizz
_buzz.c -o 9-fizz_buzz
julien@ubuntu:~/0x04$ ./9-fizz_buzz
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19 Buz
z Fizz 22 23 Fizz Buzz 26 Fizz 28 29 FizzBuzz 31 32 Fizz 34 Buzz Fizz 37 38
Fizz Buzz 41 Fizz 43 44 FizzBuzz 46 47 Fizz 49 Buzz Fizz 52 53 Fizz Buzz 56
Fizz 58 59 FizzBuzz 61 62 Fizz 64 Buzz Fizz 67 68 Fizz Buzz 71 Fizz 73 74 F
izzBuzz 76 77 Fizz 79 Buzz Fizz 82 83 Fizz Buzz 86 Fizz 88 89 FizzBuzz 91 9
2 Fizz 94 Buzz Fizz 97 98 Fizz Buzz
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `9-fizz_buzz.c`

Done! Help Check your code Get a sandbox QA Review

10. Triangles

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a triangle, followed by a new line.

- Prototype: `void print_triangle(int size);`
- You can only use `_putchar` function to print
- Where `size` is the size of the triangle
- If `size` is `0` or less, the function should print only a new line
- Use the character `#` to print the triangle

```
julien@ubuntu:~/0x04$ cat 10-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_triangle(2);
    print_triangle(10);
    print_triangle(1);
    print_triangle(0);
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 10-main.c 10-print_triangle.c -o 10-triangles
julien@ubuntu:~/0x04$ ./10-triangles
 #
##
```

```
         #
        ##
       ###
      ####
     #####
    ######
   #######
  ########
 #########
##########
#

julien@ubuntu:~/0x04$ ./10-triangles | tr ' ' . | cat -e
.#$
##$
.........#$
........##$
.......###$
......####$
.....#####$
....######$
...#######$
..########$
.#########$
##########$
#$
$
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `10-print_triangle.c`

Done! Help Check your code Get a sandbox QA Review

11. The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

The prime factors of `1231952` are `2`, `2`, `2`, `2`, `37` and `2081`.

Write a program that finds and prints the largest prime factor of the number `612852475143`, followed by a new line.

- You are allowed to use the standard library
- Your program will be compiled with this command: `gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-prime_factor.c -o 100-prime_factor -lm`

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `100-prime_factor.c`

Done! Help Check your code Get a sandbox QA Review

12. Numbers have life; they're not just symbols on paper

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints an integer.

- Prototype: `void print_number(int n);`
- You can only use `_putchar` function to print
- You are not allowed to use `long`
- You are not allowed to use arrays or pointers
- You are not allowed to hard-code special values

```
julien@ubuntu:~/0x04$ cat 101-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_number(98);
    _putchar('\n');
    print_number(402);
    _putchar('\n');
    print_number(1024);
    _putchar('\n');
    print_number(0);
    _putchar('\n');
    print_number(-98);
    _putchar('\n');
    return (0);
}
julien@ubuntu:~/0x04$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 101-main.c 101-print_number.c -o 101-print_numbers
julien@ubuntu:~/0x04$ ./101-print_numbers
98
402
1024
0
-98
julien@ubuntu:~/0x04$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x04-more_functions_nested_loops`
- File: `101-print_number.c`

0x05. C - Pointers, arrays and strings

Resources

Read or watch:

- C - Arrays
- C - Pointers
- C - Strings
- Memory Layout

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are pointers and how to use them
- What are arrays and how to use them
- What are the differences between pointers and arrays
- How to use strings and how to manipulate them
- Scope of variables

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables

- No more than 5 functions per file
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, etc… is forbidden
- You are allowed to use _putchar
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don't forget to push your header file

More Info

You do not need to learn about pointers to functions, pointers to pointers, multidimensional arrays, arrays of structures, `malloc` and `free` - yet.

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. 98 Battery st.

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that takes a pointer to an `int` as parameter and updates the value it points to to `98`.

- Prototype: `void reset_to_98(int *n);`

```
julien@ubuntu:~/0x05$ cat 0-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int n;
    n = 402;

    printf("n=%d\n", n);

    reset_to_98(&n);

    printf("n=%d\n", n);

    return (0);

}

julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-main
.c 0-reset_to_98.c -o 0-98

julien@ubuntu:~/0x05$ ./0-98

n=402

n=98

julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `0-reset_to_98.c`

Done! Help Check your code Get a sandbox QA Review

1. Don't swap horses in crossing a stream

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that swaps the values of two integers.

- Prototype: `void swap_int(int *a, int *b);`

```
julien@ubuntu:~/0x05$ cat 1-main.c

#include "main.h"

#include <stdio.h>


/**

 * main - check the code

 *

 * Return: Always 0.

 */

int main(void)
```

```
{
    int a;
    int b;

    a = 98;
    b = 42;
    printf("a=%d, b=%d\n", a, b);
    swap_int(&a, &b);
    printf("a=%d, b=%d\n", a, b);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-main
.c 1-swap.c -o 1-swap
julien@ubuntu:~/0x05$ ./1-swap
a=98, b=42
a=42, b=98
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `1-swap.c`

Done! Help Check your code QA Review

2. This report, by its very length, defends itself against the risk of being read

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that returns the length of a string.

- Prototype: `int _strlen(char *s);`

FYI: The standard library provides a similar function: `strlen`. Run `man strlen` to learn more.

```
julien@ubuntu:~/0x05$ cat 2-main.c
#include "main.h"
#include <stdio.h>

/**
```

```
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;
    int len;

    str = "My first strlen!";
    len = _strlen(str);
    printf("%d\n", len);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-main
.c 2-strlen.c -o 2-strlen
julien@ubuntu:~/0x05$ ./2-strlen
16
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `2-strlen.c`

Done! Help Check your code QA Review

3. I do not fear computers. I fear the lack of them

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a string, followed by a new line, to `stdout`.

- Prototype: `void _puts(char *str);`

FYI: The standard library provides a similar function: `puts`. Run `man puts` to learn more.

```
julien@ubuntu:~/0x05$ cat 3-main.c
#include "main.h"
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;

    str = "I do not fear computers. I fear the lack of them - Isaac Asimov"
;
    _puts(str);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 3-main.c 3-puts.c -o 3-puts
julien@ubuntu:~/0x05$ ./3-puts
I do not fear computers. I fear the lack of them - Isaac Asimov
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `3-puts.c`

Done! Help Check your code QA Review

4. I can only go one way. I've not got a reverse gear

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a string, in reverse, followed by a new line.

- Prototype: `void print_rev(char *s);`

```
julien@ubuntu:~/0x05$ cat 4-main.c
#include "main.h"
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;

    str = "I do not fear computers. I fear the lack of them - Isaac Asimov"
;
    print_rev(str);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 4-main.c 4-print_rev.c -o 4-print_rev
julien@ubuntu:~/0x05$ ./4-print_rev
vomisA caasI - meht fo kcal eht raef I .sretupmoc raef ton od I
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `4-print_rev.c`

Done! Help Check your code QA Review

5. A good engineer thinks in reverse and asks himself about the stylistic consequences of the components and systems he proposes

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that reverses a string.

- Prototype: `void rev_string(char *s);`

```
julien@ubuntu:~/0x05$ cat 5-main.c
#include "main.h"
#include <stdio.h>
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char s[10] = "My School";

    printf("%s\n", s);
    rev_string(s);
    printf("%s\n", s);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5-main
.c 5-rev_string.c -o 5-rev_string
julien@ubuntu:~/0x05$ ./5-rev_string
My School
loohcS yM
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `5-rev_string.c`

Done! Help Check your code QA Review

6. Half the lies they tell about me aren't true

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints every other character of a string, starting with the first character, followed by a new line.

- Prototype: `void puts2(char *str);`

```
julien@ubuntu:~/0x05$ cat 6-main.c
#include "main.h"
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;

    str = "0123456789";
    puts2(str);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 6-main.c 6-puts2.c -o 6-puts2
julien@ubuntu:~/0x05$ ./6-puts2
02468
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `6-puts2.c`

Done! Help Check your code QA Review

7. Winning is only half of it. Having fun is the other half

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints half of a string, followed by a new line.

- Prototype: `void puts_half(char *str);`
- The function should print the second half of the string
- If the number of characters is odd, the function should print the last `n` characters of the string, where `n = (length_of_the_string - 1) / 2`

```
julien@ubuntu:~/0x05$ cat 7-main.c
#include "main.h"
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;

    str = "0123456789";
    puts_half(str);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 7-main.c 7-puts_half.c -o 7-puts_half
julien@ubuntu:~/0x05$ ./7-puts_half
56789
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `7-puts_half.c`

Done! Help Check your code QA Review

8. Arrays are not pointers

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints `n` elements of an array of integers, followed by a new line.

- Prototype: `void print_array(int *a, int n);`
- where `n` is the number of elements of the array to be printed
- Numbers must be separated by comma, followed by a space
- The numbers should be displayed in the same order as they are stored in the array
- You are allowed to use `printf`

```
julien@ubuntu:~/0x05$ cat 8-main.c
#include "main.h"
```

```
/**
 * main - check the code for
 *
 * Return: Always 0.
 */
int main(void)
{
    int array[5];

    array[0] = 98;
    array[1] = 402;
    array[2] = -198;
    array[3] = 298;
    array[4] = -1024;
    print_array(array, 5);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 8-main
.c 8-print_array.c -o 8-print_array
julien@ubuntu:~/0x05$ ./8-print_array
98, 402, -198, 298, -1024
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `8-print_array.c`

Done! Help Check your code QA Review

9. strcpy

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

- Prototype: `char *_strcpy(char *dest, char *src);`

Write a function that copies the string pointed to by `src`, including the terminating null byte (`\0`), to the buffer pointed to by `dest`.

- Return value: the pointer to `dest`

FYI: The standard library provides a similar function: `strcpy`. Run `man strcpy` to learn more.

```
julien@ubuntu:~/0x05$ cat 9-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char s1[98];
    char *ptr;

    ptr = _strcpy(s1, "First, solve the problem. Then, write the code\n");
    printf("%s", s1);
    printf("%s", ptr);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 9-main
.c 9-strcpy.c -o 9-strcpy
julien@ubuntu:~/0x05$ ./9-strcpy
First, solve the problem. Then, write the code
First, solve the problem. Then, write the code
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `9-strcpy.c`

Done! Help Check your code QA Review

10. Great leaders are willing to sacrifice the numbers to save the people. Poor leaders sacrifice the people to save the numbers

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that convert a string to an integer.

- Prototype: `int _atoi(char *s);`
- The number in the string can be preceded by an infinite number of characters
- You need to take into account all the `-` and `+` signs before the number
- If there are no numbers in the string, the function must return `0`
- You are not allowed to use `long`
- You are not allowed to declare new variables of “type” array
- You are not allowed to hard-code special values
- We will use the `-fsanitize=signed-integer-overflow` gcc flag to compile your code.

FYI: The standard library provides a similar function: `atoi`. Run `man atoi` to learn more.

```
julien@ubuntu:~/0x05$ cat 100-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int nb;

    nb = _atoi("98");
    printf("%d\n", nb);
    nb = _atoi("-402");
    printf("%d\n", nb);
    nb = _atoi("          ------++++++-----+++++--98");
    printf("%d\n", nb);
    nb = _atoi("214748364");
    printf("%d\n", nb);
    nb = _atoi("0");
    printf("%d\n", nb);
    nb = _atoi("Suite 402");
    printf("%d\n", nb);
    nb = _atoi("         +      +    -    -98 Battery Street; San Francisco
, CA 94111 - USA             ");
    printf("%d\n", nb);
    nb = _atoi("---++++ -++ Sui - te -   402 #cisfun :)");
    printf("%d\n", nb);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 -fsanitize=signed-integer-overflow 100-main.c 100-atoi.c -o 100-atoi
julien@ubuntu:~/0x05$ ./100-atoi
98
-402
-98
214748364
0
402
98
402
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `100-atoi.c`

Done! Help Check your code QA Review

11. Don't hate the hacker, hate the code

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Create a program that generates random valid passwords for the program 101-crackme.

- You are allowed to use the standard library
- You don't have to pass the `betty-style` tests (you still need to pass the `betty-doc` tests)
- man `srand`, `rand`, `time`
- `gdb` and `objdump` can help

```
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra 101-keygen.c -o 1
01-keygen

julien@ubuntu:~/0x05$ ./101-crackme "`./101-keygen`"

Tada! Congrats

julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `101-keygen.c`

0x06. C - More pointers, arrays and strings

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are pointers and how to use them
- What are arrays and how to use them
- What are the differences between pointers and arrays
- How to use strings and how to manipulate them
- Scope of variables

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, etc… is forbidden
- You are allowed to use _putchar
- You don’t have to push `_putchar.c`, we will use our file. If you do it won’t be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don’t have to push them to your repo (if you do we won’t take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples

- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don’t forget to push your header file

Quiz questions
Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. strcat

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that concatenates two strings.

- Prototype: `char *_strcat(char *dest, char *src);`
- This function appends the `src` string to the `dest` string, overwriting the terminating null byte (`\0`) at the end of `dest`, and then adds a terminating null byte
- Returns a pointer to the resulting string `dest`

FYI: The standard library provides a similar function: `strcat`. Run `man strcat` to learn more.

```
julien@ubuntu:~/0x06$ cat 0-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char s1[98] = "Hello ";
    char s2[] = "World!\n";
    char *ptr;

    printf("%s\n", s1);
    printf("%s", s2);
    ptr = _strcat(s1, s2);
    printf("%s", s1);
    printf("%s", s2);
    printf("%s", ptr);
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-main
.c 0-strcat.c -o 0-strcat
julien@ubuntu:~/0x06$ ./0-strcat
Hello
World!
Hello World!
World!
Hello World!
julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `0-strcat.c`

Done! Help Check your code Get a sandbox QA Review

1. strncat

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that concatenates two strings.

- Prototype: `char *_strncat(char *dest, char *src, int n);`
- The `_strncat` function is similar to the `_strcat` function, except that
  - it will use at most `n` bytes from `src`; and
  - `src` does not need to be null-terminated if it contains `n` or more bytes
- Return a pointer to the resulting string `dest`

FYI: The standard library provides a similar function: `strncat`. Run `man strncat` to learn more.

```
julien@ubuntu:~/0x06$ cat 1-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char s1[98] = "Hello ";
    char s2[] = "World!\n";
    char *ptr;

    printf("%s\n", s1);
    printf("%s", s2);
    ptr = _strncat(s1, s2, 1);
    printf("%s\n", s1);
    printf("%s", s2);
    printf("%s\n", ptr);
    ptr = _strncat(s1, s2, 1024);
    printf("%s", s1);
    printf("%s", s2);
    printf("%s", ptr);
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-main
.c 1-strncat.c -o 1-strncat
julien@ubuntu:~/0x06$ ./1-strncat
Hello
World!
Hello W
World!
Hello W
Hello WWorld!
World!
Hello WWorld!
julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `1-strncat.c`

Done! Help Check your code Get a sandbox QA Review

2. strncpy

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that copies a string.

- Prototype: `char *_strncpy(char *dest, char *src, int n);`
- Your function should work exactly like `strncpy`

FYI: The standard library provides a similar function: `strncpy`. Run `man strncpy` to learn more.

```
julien@ubuntu:~/0x06$ cat 2-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char s1[98];
    char *ptr;
    int i;

    for (i = 0; i < 98 - 1; i++)
    {
        s1[i] = '*';
    }
    s1[i] = '\0';
    printf("%s\n", s1);
    ptr = _strncpy(s1, "First, solve the problem. Then, write the code\n",
5);
```

```
    printf("%s\n", s1);
    printf("%s\n", ptr);
    ptr = _strncpy(s1, "First, solve the problem. Then, write the code\n",
90);
    printf("%s", s1);
    printf("%s", ptr);
    for (i = 0; i < 98; i++)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", s1[i]);
    }
    printf("\n");
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-main
.c 2-strncpy.c -o 2-strncpy
julien@ubuntu:~/0x06$ ./2-strncpy
****************************************************************************
**********************
First***********************************************************************
**********************
First***********************************************************************
**********************
First, solve the problem. Then, write the code
First, solve the problem. Then, write the code
0x46 0x69 0x72 0x73 0x74 0x2c 0x20 0x73 0x6f 0x6c
0x76 0x65 0x20 0x74 0x68 0x65 0x20 0x70 0x72 0x6f
0x62 0x6c 0x65 0x6d 0x2e 0x20 0x54 0x68 0x65 0x6e
```

```
0x2c 0x20 0x77 0x72 0x69 0x74 0x65 0x20 0x74 0x68
0x65 0x20 0x63 0x6f 0x64 0x65 0x0a 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x2a 0x2a 0x2a 0x2a 0x2a 0x2a 0x2a 0x00
julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `2-strncpy.c`

Done! Help Check your code Get a sandbox QA Review

3. strcmp

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that compares two strings.

- Prototype: `int _strcmp(char *s1, char *s2);`
- Your function should work exactly like `strcmp`

FYI: The standard library provides a similar function: `strcmp`. Run `man strcmp` to learn more.

```
julien@ubuntu:~/0x06$ cat 3-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char s1[] = "Hello";
    char s2[] = "World!";
    printf("%d\n", _strcmp(s1, s2));
    printf("%d\n", _strcmp(s2, s1));
    printf("%d\n", _strcmp(s1, s1));
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-main
.c 3-strcmp.c -o 3-strcmp
julien@ubuntu:~/0x06$ ./3-strcmp
-15
15
0
julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `3-strcmp.c`

Done! Help Check your code Get a sandbox QA Review

4. I am a kind of paranoid in reverse. I suspect people of plotting to make me happy

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that reverses the content of an array of integers.

- Prototype: `void reverse_array(int *a, int n);`
- Where `n` is the number of elements of the array

```
julien@ubuntu:~/0x06$ cat 4-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 * @a: an array of integers
 * @n: the number of elements to swap
 *
 * Return: nothing.
 */
void print_array(int *a, int n)
{
    int i;

    i = 0;
    while (i < n)
    {
        if (i != 0)
        {
            printf(", ");
        }
        printf("%d", a[i]);
        i++;
    }
    printf("\n");
}

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int a[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 98, 1024, 1337};

    print_array(a, sizeof(a) / sizeof(int));
    reverse_array(a, sizeof(a) / sizeof(int));
    print_array(a, sizeof(a) / sizeof(int));
    return (0);
}
```

```
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-main
.c 4-rev_array.c -o 4-rev_array

julien@ubuntu:~/0x06$ ./4-rev_array

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 98, 1024, 1337

1337, 1024, 98, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `4-rev_array.c`

Done! Help Check your code Get a sandbox QA Review

5. Always look up

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that changes all lowercase letters of a string to uppercase.

- Prototype: `char *string_toupper(char *);`

```
julien@ubuntu:~/0x06$ cat 5-main.c

#include "main.h"

#include <stdio.h>


/**

 * main - check the code

 *

 * Return: Always 0.

 */

int main(void)

{

    char str[] = "Look up!\n";

    char *ptr;


    ptr = string_toupper(str);

    printf("%s", ptr);

    printf("%s", str);
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5-main
.c 5-string_toupper.c -o 5-string_toupper
julien@ubuntu:~/0x06$ ./5-string_toupper
LOOK UP!
LOOK UP!
julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `5-string_toupper.c`

Done! Help Check your code Get a sandbox QA Review

6. Expect the best. Prepare for the worst. Capitalize on what comes

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that capitalizes all words of a string.

- Prototype: `char *cap_string(char *);`
- Separators of words: space, tabulation, new line, `,`, `;`, `.`, `!`, `?`, `"`, `(`, `)`, `{`, and `}`

```
julien@ubuntu:~/0x06$ cat 6-main.c
#include "main.h"
#include <stdio.h>


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char str[] = "Expect the best. Prepare for the worst. Capitalize on wha
t comes.\nhello world! hello-world 0123456hello world\thello world.hello wo
rld\n";
    char *ptr;
    ptr = cap_string(str);
    printf("%s", ptr);
    printf("%s", str);
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 6-main
.c 6-cap_string.c -o 6-cap
julien@ubuntu:~/0x06$ ./6-cap
Expect The Best. Prepare For The Worst. Capitalize On What Comes.
Hello World! Hello-world 0123456hello World Hello World.Hello World
Expect The Best. Prepare For The Worst. Capitalize On What Comes.
Hello World! Hello-world 0123456hello World Hello World.Hello World
julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `6-cap_string.c`

Done! Help Check your code Get a sandbox QA Review

7. Mozart composed his music not for the elite, but for everybody

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that encodes a string into 1337.

- Letters `a` and `A` should be replaced by `4`
- Letters `e` and `E` should be replaced by `3`
- Letters `o` and `O` should be replaced by `0`
- Letters `t` and `T` should be replaced by `7`
- Letters `l` and `L` should be replaced by `1`
- Prototype: `char *leet(char *);`
- You can only use one `if` in your code
- You can only use two loops in your code
- You are not allowed to use `switch`
- You are not allowed to use any ternary operation

```
julien@ubuntu:~/0x06$ cat 7-main.c
#include "main.h"
#include <stdio.h>
```

```
/**
 * main - check the code for
 *
 * Return: Always 0.
 */
int main(void)
{
    char s[] = "Expect the best. Prepare for the worst. Capitalize on what comes.\n";
    char *p;

    p = leet(s);
    printf("%s", p);
    printf("%s", s);
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 7-main.c 7-leet.c -o 7-1337
julien@ubuntu:~/0x06$ ./7-1337
3xp3c7 7h3 b3s7. Pr3p4r3 f0r 7h3 w0rs7. C4pi741iz3 0n wh47 c0m3s.
3xp3c7 7h3 b3s7. Pr3p4r3 f0r 7h3 w0rs7. C4pi741iz3 0n wh47 c0m3s.
julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `7-leet.c`

Done! Help Check your code Get a sandbox QA Review

8. rot13

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that encodes a string using rot13.

- Prototype: `char *rot13(char *);`
- You can only use `if` statement once in your code
- You can only use two loops in your code
- You are not allowed to use `switch`
- You are not allowed to use any ternary operation

```
julien@ubuntu:~/0x06$ cat 100-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char s[] = "ROT13 (\"rotate by 13 places\", sometimes hyphenated ROT-13
) is a simple letter substitution cipher.\n";
    char *p;

    p = rot13(s);
    printf("%s", p);
    printf("------------------------------------\n");
    printf("%s", s);
    printf("------------------------------------\n");
    p = rot13(s);
    printf("%s", p);
    printf("------------------------------------\n");
    printf("%s", s);
    printf("------------------------------------\n");
    p = rot13(s);
    printf("%s", p);
    printf("------------------------------------\n");
    printf("%s", s);
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-ma
in.c 100-rot13.c -o 100-rot13
julien@ubuntu:~/0x06$ ./100-rot13
```

```
EBG13 ("ebgngr ol 13 cynprf", fbzrgvzrf ulcurangrq EBG-13) vf n fvzcyr yrgg
re fhofgvghgvba pvcure.

------------------------------------

EBG13 ("ebgngr ol 13 cynprf", fbzrgvzrf ulcurangrq EBG-13) vf n fvzcyr yrgg
re fhofgvghgvba pvcure.

------------------------------------

ROT13 ("rotate by 13 places", sometimes hyphenated ROT-13) is a simple lett
er substitution cipher.

------------------------------------

ROT13 ("rotate by 13 places", sometimes hyphenated ROT-13) is a simple lett
er substitution cipher.

------------------------------------

EBG13 ("ebgngr ol 13 cynprf", fbzrgvzrf ulcurangrq EBG-13) vf n fvzcyr yrgg
re fhofgvghgvba pvcure.

------------------------------------

EBG13 ("ebgngr ol 13 cynprf", fbzrgvzrf ulcurangrq EBG-13) vf n fvzcyr yrgg
re fhofgvghgvba pvcure.

julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `100-rot13.c`

Done! Help Check your code Get a sandbox QA Review

9. Numbers have life; they're not just symbols on paper

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints an integer.

- Prototype: `void print_number(int n);`
- You can only use `_putchar` function to print
- You are not allowed to use `long`
- You are not allowed to use arrays or pointers
- You are not allowed to hard-code special values

```
julien@ubuntu:~/0x06$ cat 101-main.c

#include "main.h"



/**
```

```
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_number(98);
    _putchar('\n');
    print_number(402);
    _putchar('\n');
    print_number(1024);
    _putchar('\n');
    print_number(0);
    _putchar('\n');
    print_number(-98);
    _putchar('\n');
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putch
ar.c 101-main.c 101-print_number.c -o 101-print_numbers
julien@ubuntu:~/0x06$ ./101-print_numbers
98
402
1024
0
-98
julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `101-print_number.c`

Done! Help Check your code Get a sandbox QA Review

10. A dream doesn't become reality through magic; it takes sweat, determination and hard work

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Add one line to this code, so that the program prints `a[2] = 98`, followed by a new line.

- You are not allowed to use the variable `a` in your new line of code
- You are not allowed to modify the variable `p`
- You can only write one statement
- You are not allowed to use `,`
- You are not allowed to code anything else than the line of expected line of code at the expected line
- Your code should be written at line 19, before the `;`
- Do not remove anything from the initial code (not even the comments)
- and don't change anything but the line of code you are adding (don't change the spaces to tabs!)
- You are allowed to use the standard library

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `102-magic.c`

Done! Help Check your code Get a sandbox QA Review

11. It is the addition of strangeness to beauty that constitutes the romantic character in art

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that adds two numbers.

- Prototype: `char *infinite_add(char *n1, char *n2, char *r, int size_r);`
- Where `n1` and `n2` are the two numbers
- `r` is the buffer that the function will use to store the result
- `size_r` is the buffer size
- The function returns a pointer to the result
- You can assume that you will always get positive numbers, or `0`
- You can assume that there will be only digits in the strings `n1` and `n2`
- `n1` and `n2` will never be empty
- If the result can not be stored in `r` the function must return `0`

```
julien@ubuntu:~/0x06$ cat 103-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
        char *n = "12345678924345743678235745756784776857856456858768767745
86734734563456453743756756784458";
        char *m = "90347906634706972346829145693462596349586932465973246597
62347956349265983465962349569346";
        char r[100];
        char r2[10];
        char r3[11];
        char *res;

        res = infinite_add(n, m, r, 100);
        if (res == 0)
        {
                printf("Error\n");
        }
        else
        {
                printf("%s + %s = %s\n", n, m, res);
        }
        n = "1234567890";
        m = "1";
        res = infinite_add(n, m, r2, 10);
        if (res == 0)
        {
                printf("Error\n");
        }
        else
        {
                printf("%s + %s = %s\n", n, m, res);
        }
        n = "999999999";
        m = "1";
        res = infinite_add(n, m, r2, 10);
        if (res == 0)
        {
                printf("Error\n");
        }
        else
        {
                printf("%s + %s = %s\n", n, m, res);
        }
        res = infinite_add(n, m, r3, 11);
        if (res == 0)
        {
                printf("Error\n");
        }
        else
        {
                printf("%s + %s = %s\n", n, m, res);
        }
        return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 103-ma
in.c 103-infinite_add.c -o 103-add
julien@ubuntu:~/0x06$ ./103-add
123456789243457436782357457567847768578564568587687677458673473456345645374
3756756784458 + 903479066347069723468291456934625963495869324659732465976234
79563492659834659623495693 46 = 1026935855590527160250648914502473732074433
89324742014343490826909127224372097191063538 04
Error
Error
999999999 + 1 = 1000000000
julien@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `103-infinite_add.c`

Done! Help Check your code Get a sandbox QA Review

12. Noise is a buffer, more effective than cubicles or booth walls

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a buffer.

- Prototype: `void print_buffer(char *b, int size);`
- The function must print the content of `size` bytes of the buffer pointed by `b`
- The output should print 10 bytes per line
- Each line starts with the position of the first byte of the line in hexadecimal (8 chars), starting with `0`
- Each line shows the hexadecimal content (2 chars) of the buffer, 2 bytes at a time, separated by a space
- Each line shows the content of the buffer. If the byte is a printable character, print the letter, if not, print `.`
- Each line ends with a new line `\n`
- If `size` is 0 or less, the output should be a new line only `\n`
- You are allowed to use the standard library
- The output should look like the following example, and formatted exactly the same way:

```
julien@ubuntu:~/0x06$ cat 104-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char buffer[] = "This is a string!\0And this is the rest of the #buffer :)\1\2\3\4\5\6\7#cisfun\n\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\x20\x21\x34\x56#pointersarefun #infernumisfun\n";

    printf("%s\n", buffer);
    printf("---------------------------------\n");
    print_buffer(buffer, sizeof(buffer));
    return (0);
}
julien@ubuntu:~/0x06$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 104-main.c 104-print_buffer.c -o 104-buffer
julien@ubuntu:~/0x06$ ./104-buffer
This is a string!
---------------------------------
00000000: 5468 6973 2069 7320 6120 This is a 
0000000a: 7374 7269 6e67 2100 416e string!.An
00000014: 6420 7468 6973 2069 7320 d this is 
0000001e: 7468 6520 7265 7374 206f the rest o
00000028: 6620 7468 6520 2362 7566 f the #buf
00000032: 6665 7220 3a29 0102 0304 fer :)....
0000003c: 0506 0723 6369 7366 756e ...#cisfun
00000046: 0a00 0000 0000 0000 0000 ..........
00000050: 0000 0000 0000 0000 0000 ..........
0000005a: 2021 3456 2370 6f69 6e74  !4V#point
00000064: 6572 7361 7265 6675 6e20 ersarefun 
0000006e: 2369 6e66 6572 6e75 6d69 #infernumi
00000078: 7366 756e 0a00           sfun..
julien@ubuntu:~/0x06$
```

Repo:

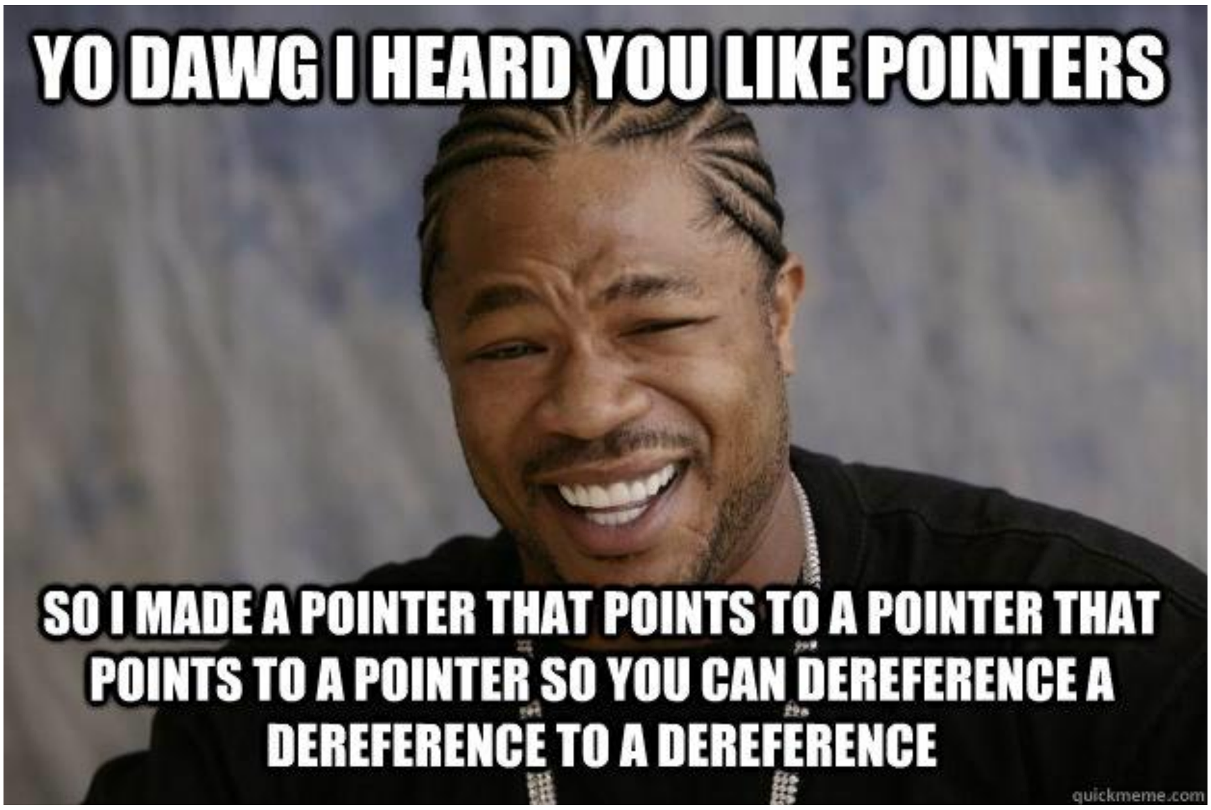
- GitHub repository: `alx-low_level_programming`
- Directory: `0x06-pointers_arrays_strings`
- File: `104-print_buffer.c`

0x07. C - Even more pointers, arrays and strings

Concepts

For this project, we expect you to look at this concept:

- Pointers and arrays


YO DAWG I HEARD YOU LIKE POINTERS
SO I MADE A POINTER THAT POINTS TO A POINTER THAT POINTS TO A POINTER SO YOU CAN DEREFERENCE A DEREFERENCE TO A DEREFERENCE
quickmeme.com

Resources

Read or watch:

- C - Pointer to Pointer
- C – Pointer to Pointer with example
- Multi-dimensional Arrays in C
- Two dimensional (2D) arrays in C programming with example

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are pointers to pointers and how to use them
- What are multidimensional arrays and how to use them
- What are the most common C standard library functions to manipulate strings

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, etc… is forbidden
- You are allowed to use _putchar
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don't forget to push your header file

More Info

You do not need to learn about pointers to functions, arrays of structures, `malloc` and `free` - yet.

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. memset

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that fills memory with a constant byte.

- Prototype: `char *_memset(char *s, char b, unsigned int n);`
- The `_memset()` function fills the first `n` bytes of the memory area pointed to by `s` with the constant byte `b`
- Returns a pointer to the memory area `s`

FYI: The standard library provides a similar function: `memset`. Run `man memset` to learn more.

```
julien@ubuntu:~/0x07$ cat 0-main.c
#include "main.h"
#include <stdio.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(char *buffer, unsigned int size)
{
        unsigned int i;

        i = 0;
        while (i < size)
        {
                if (i % 10)
                {
                        printf(" ");
                }
                if (!(i % 10) && i)
                {
                        printf("\n");
                }
                printf("0x%02x", buffer[i]);
                i++;
        }
        printf("\n");
}

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char buffer[98] = {0x00};

    simple_print_buffer(buffer, 98);
    _memset(buffer, 0x01, 95);
    printf("-------------------------------------------------\n");
    simple_print_buffer(buffer, 98);
    return (0);
}
julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-main
.c 0-memset.c -o 0-memset
julien@ubuntu:~/0x07$ ./0-memset
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

```
-------------------------------------------------
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x00 0x00 0x00
julien@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`
- File: `0-memset.c`

Done! Help Check your code Get a sandbox QA Review

1. memcpy

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that copies memory area.

- Prototype: `char *_memcpy(char *dest, char *src, unsigned int n);`
- The `_memcpy()` function copies `n` bytes from memory area `src` to memory area `dest`
- Returns a pointer to `dest`

FYI: The standard library provides a similar function: `memcpy`. Run `man memcpy` to learn more.

```
julien@ubuntu:~/0x07$ cat 1-main.c
#include "main.h"
#include <stdio.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(char *buffer, unsigned int size)
{
    unsigned int i;

    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", buffer[i]);
        i++;
    }
    printf("\n");
}

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char buffer[98] = {0};
    char buffer2[98] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14};

    simple_print_buffer(buffer, 98);
    _memcpy(buffer + 50, buffer2, 10);
    printf("-------------------------------------------------\n");
    simple_print_buffer(buffer, 98);
    return (0);
}
julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-main.c 1-memcpy.c -o 1-memcpy
julien@ubuntu:~/0x07$ ./1-memcpy
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
-------------------------------------------------
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x01 0x02 0x03 0x04 0x05 0x07 0x07 0x08 0x09 0x0a
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
julien@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`

- File: `1-memcpy.c`

Done! Help Check your code Get a sandbox QA Review

2. strchr

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that locates a character in a string.

- Prototype: `char *_strchr(char *s, char c);`
- Returns a pointer to the first occurrence of the character `c` in the string `s`, or `NULL` if the character is not found

FYI: The standard library provides a similar function: `strchr`. Run `man strchr` to learn more.

```
julien@ubuntu:~/0x07$ cat 2-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *s = "hello";
    char *f;

    f = _strchr(s, 'l');

    if (f != NULL)
    {
        printf("%s\n", f);
    }
    return (0);
}
julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-main
.c 2-strchr.c -o 2-strchr
```

```
julien@ubuntu:~/0x07$ ./2-strchr
llo
julien@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`
- File: `2-strchr.c`

Done! Help Check your code Get a sandbox QA Review

3. strspn

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that gets the length of a prefix substring.

- Prototype: `unsigned int _strspn(char *s, char *accept);`
- Returns the number of bytes in the initial segment of `s` which consist only of bytes from `accept`

FYI: The standard library provides a similar function: `strspn`. Run `man strspn` to learn more.

```
julien@ubuntu:~/0x07$ cat 3-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *s = "hello, world";
    char *f = "oleh";
    unsigned int n;

    n = _strspn(s, f);
    printf("%u\n", n);
    return (0);
}
julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-main
.c 3-strspn.c -o 3-strspn
julien@ubuntu:~/0x07$ ./3-strspn
5
julien@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`
- File: `3-strspn.c`

Done! Help Check your code Get a sandbox QA Review

4. strpbrk

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that searches a string for any of a set of bytes.

- Prototype: `char *_strpbrk(char *s, char *accept);`
- The `_strpbrk()` function locates the first occurrence in the string `s` of any of the bytes in the string `accept`
- Returns a pointer to the byte in `s` that matches one of the bytes in `accept`, or `NULL` if no such byte is found

FYI: The standard library provides a similar function: `strpbrk`. Run `man strpbrk` to learn more.

```
julien@ubuntu:~/0x07$ cat 4-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *s = "hello, world";
    char *f = "world";
    char *t;
    t = _strpbrk(s, f);
    printf("%s\n", t);
    return (0);
}
julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-main
.c 4-strpbrk.c -o 4-strpbrk
julien@ubuntu:~/0x07$ ./4-strpbrk
llo, world
julien@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`
- File: `4-strpbrk.c`

Done! Help Check your code Get a sandbox QA Review

5. strstr

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that locates a substring.

- Prototype: `char *_strstr(char *haystack, char *needle);`
- The `_strstr()` function finds the first occurrence of the substring `needle` in the string `haystack`. The terminating null bytes (`\0`) are not compared
- Returns a pointer to the beginning of the located substring, or `NULL` if the substring is not found.

FYI: The standard library provides a similar function: `strstr`. Run `man strstr` to learn more.

```
julien@ubuntu:~/0x07$ cat 5-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
```

```
{
    char *s = "hello, world";
    char *f = "world";
    char *t;

    t = _strstr(s, f);
    printf("%s\n", t);
    return (0);
}
julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5-main
.c 5-strstr.c -o 5-strstr
julien@ubuntu:~/0x07$ ./5-strstr
world
julien@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`
- File: `5-strstr.c`

Done! Help Check your code Get a sandbox QA Review

6. Chess is mental torture

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the chessboard.

- Prototype: `void print_chessboard(char (*a)[8]);`

```
julien@ubuntu:~/0x07$ cat 7-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char board[8][8] = {
        {'r', 'k', 'b', 'q', 'k', 'b', 'k', 'r'},
        {'p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'},
        {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},
        {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},
        {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},
        {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},
        {'P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'},
        {'R', 'K', 'B', 'Q', 'K', 'B', 'K', 'R'},
    };
    print_chessboard(board);
    return (0);
}
julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 7-main.c 7-print_chessboard.c -o 7-print_chessboard
julien@ubuntu:~/0x07$ ./7-print_chessboard
rkbqkbkr
pppppppp
        
        
        
        
PPPPPPPP
RKBQKBKR
julien@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`
- File: `7-print_chessboard.c`

Done! Help Check your code Get a sandbox QA Review

7. The line of life is a ragged diagonal between duty and desire

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the sum of the two diagonals of a square matrix of integers.

- Prototype: `void print_diagsums(int *a, int size);`
- Format: see example
- You are allowed to use the standard library

Note that in the following example we are casting an `int[][]` into an `int*`. This is not something you should do. The goal here is to make sure you understand how an array of array is stored in memory.

```
julien@ubuntu:~/0x07$ cat 8-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int c3[3][3] = {
        {0, 1, 5},
        {10, 11, 12},
        {1000, 101, 102},
    };
    int c5[5][5] = {
        {0, 1, 5, 12124, 1234},
        {10, 11, 12, 123521, 12512},
        {1000, 101, 102, 12545, 214543435},
        {100, 1012451, 11102, 12545, 214543435},
        {10, 12401, 10452, 11542545, 1214543435},
    };
    print_diagsums((int *)c3, 3);
    print_diagsums((int *)c5, 5);
    return (0);
}
julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 8-main
.c 8-print_diagsums.c -o 8-print_diagsums
julien@ubuntu:~/0x07$ ./8-print_diagsums
113, 1016
1214556093, 1137318
julien@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`
- File: `8-print_diagsums.c`

Done! Help Check your code Get a sandbox QA Review

8. Double pointer, double fun

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that sets the value of a pointer to a char.

- Prototype: `void set_string(char **s, char *to);`

```
julien@ubuntu:~/0x07$ cat 100-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *s0 = "Bob Dylan";
    char *s1 = "Robert Allen";

    printf("%s, %s\n", s0, s1);
    set_string(&s1, s0);
    printf("%s, %s\n", s0, s1);
    return (0);
}
julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-ma
in.c 100-set_string.c -o 100-set_string
julien@ubuntu:~/0x07$ ./100-set_string
Bob Dylan, Robert Allen
Bob Dylan, Bob Dylan
julien@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`
- File: `100-set_string.c`

Done! Help Check your code QA Review

9. My primary goal of hacking was the intellectual curiosity, the seduction of adventure

#advanced

Score: 0.0% (*Checks completed: 0.0%*)

Create a file that contains the password for the crackme2 executable.

- Your file should contain the exact password, no new line, no extra space
- `ltrace`, `ldd`, `gdb` and `objdump` can help
- You may need to install the `openssl` library to run the `crakme2` program: `sudo apt install libssl-dev`
- Edit the source list `sudo nano /etc/apt/sources.list` to add the following line: `deb http://security.ubuntu.com/ubuntu xenial-security main` Then `sudo apt update` and `sudo apt install libssl1.0.0`

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x07-pointers_arrays_strings`
- File: `101-crackme_password`

0x08. C - Recursion

Resources

Read or watch:

- 0x08. Recursion, introduction
- What on Earth is Recursion?
- C - Recursion
- C Programming Tutorial 85, Recursion pt.1
- C Programming Tutorial 86, Recursion pt.2

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What is recursion
- How to implement recursion
- In what situations you should implement recursion
- In what situations you shouldn't implement recursion

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, etc… is forbidden

- You are allowed to use _putchar
- You don’t have to push `_putchar.c`, we will use our file. If you do it won’t be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don’t have to push them to your repo (if you do we won’t take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don’t forget to push your header file
- **You are not allowed to use any kind of loops**
- You are not allowed to use `static` variables

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. She locked away a secret, deep inside herself, something she once knew to be true... but chose to forget

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a string, followed by a new line.

- Prototype: `void _puts_recursion(char *s);`

FYI: The standard library provides a similar function: `puts`. Run `man puts` to learn more.

```
julien@ubuntu:~/0x08. Recursion$ cat 0-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    _puts_recursion("Puts with recursion");
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 0-main.c 0-puts_recursion.c -o 0-puts_recursion
```

```
julien@ubuntu:~/0x08. Recursion$ ./0-puts_recursion
Puts with recursion
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `0-puts_recursion.c`

Done! Help Check your code Get a sandbox QA Review

1. Why is it so important to dream? Because, in my dreams we are together

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a string in reverse.

- Prototype: `void _print_rev_recursion(char *s);`

```
julien@ubuntu:~/0x08. Recursion$ cat 1-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    _print_rev_recursion("\nColton Walker");
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 1-main.c 1-print_rev_recursion.c -o 1-print_rev_recursion
julien@ubuntu:~/0x08. Recursion$ ./1-print_rev_recursion
reklaW notloC
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `1-print_rev_recursion.c`

Done! Help Check your code Get a sandbox QA Review

2. Dreams feel real while we're in them. It's only when we wake up that we realize something was actually strange

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that returns the length of a string.

- Prototype: `int _strlen_recursion(char *s);`

FYI: The standard library provides a similar function: `strlen`. Run `man strlen` to learn more.

```
julien@ubuntu:~/0x08. Recursion$ cat 2-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int n;

    n = _strlen_recursion("Corbin Coleman");
    printf("%d\n", n);
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89  2-main.c 2-strlen_recursion.c -o 2-strlen_recursion
julien@ubuntu:~/0x08. Recursion$ ./2-strlen_recursion
14
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `2-strlen_recursion.c`

Done! Help Check your code Get a sandbox QA Review

3. You mustn't be afraid to dream a little bigger, darling

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that returns the factorial of a given number.

- Prototype: `int factorial(int n);`
- If `n` is lower than `0`, the function should return `-1` to indicate an error
- Factorial of `0` is `1`

```
julien@ubuntu:~/0x08. Recursion$ cat 3-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int r;

    r = factorial(1);
    printf("%d\n", r);
    r = factorial(5);
    printf("%d\n", r);
    r = factorial(10);
    printf("%d\n", r);
    r = factorial(-1024);
    printf("%d\n", r);
    return (0);
}
```

```
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-main.c 3-factorial.c -o 3-factorial

julien@ubuntu:~/0x08. Recursion$ ./3-factorial

1

120

3628800

-1

julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `3-factorial.c`

Done! Help Check your code Get a sandbox QA Review

4. Once an idea has taken hold of the brain it's almost impossible to eradicate

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that returns the value of `x` raised to the power of `y`.

- Prototype: `int _pow_recursion(int x, int y);`
- If `y` is lower than `0`, the function should return `-1`

FYI: The standard library provides a different function: `pow`. Run `man pow` to learn more.

```
julien@ubuntu:~/0x08. Recursion$ cat 4-main.c

#include "main.h"

#include <stdio.h>


/**

 * main - check the code

 *

 * Return: Always 0.

 */

int main(void)

{

    int r;
    r = _pow_recursion(1, 10);
    printf("%d\n", r);
    r = _pow_recursion(1024, 0);
    printf("%d\n", r);
    r = _pow_recursion(2, 16);
    printf("%d\n", r);
    r = _pow_recursion(5, 2);
    printf("%d\n", r);
    r = _pow_recursion(5, -2);
    printf("%d\n", r);
    r = _pow_recursion(-5, 3);
    printf("%d\n", r);
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-main.c 4-pow_recursion.c -o 4-pow
julien@ubuntu:~/0x08. Recursion$ ./4-pow
1
1
65536
25
-1
-125
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `4-pow_recursion.c`

Done! Help Check your code Get a sandbox QA Review

5. Your subconscious is looking for the dreamer

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that returns the natural square root of a number.

- Prototype: `int _sqrt_recursion(int n);`

- If `n` does not have a natural square root, the function should return `-1`

FYI: The standard library provides a different function: `sqrt`. Run `man sqrt` to learn more.

```
julien@ubuntu:~/0x08. Recursion$ cat 5-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int r;

    r = _sqrt_recursion(1);
    printf("%d\n", r);
    r = _sqrt_recursion(1024);
    printf("%d\n", r);
    r = _sqrt_recursion(16);
    printf("%d\n", r);
    r = _sqrt_recursion(17);
    printf("%d\n", r);
    r = _sqrt_recursion(25);
    printf("%d\n", r);
    r = _sqrt_recursion(-1);
    printf("%d\n", r);
    return (0);
}
julien@ubuntu:~/0x08. gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5-main
.c 5-sqrt_recursion.c -o 5-sqrt
julien@ubuntu:~/0x08. Recursion$ ./5-sqrt
1
```

```
32
4
-1
5
-1
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `5-sqrt_recursion.c`

Done! Help Check your code Get a sandbox QA Review

6. Inception. Is it possible?

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that returns `1` if the input integer is a prime number, otherwise return `0`.

- Prototype: `int is_prime_number(int n);`

```
julien@ubuntu:~/0x08. Recursion$ cat 6-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int r;

    r = is_prime_number(1);
    printf("%d\n", r);
    r = is_prime_number(1024);
    printf("%d\n", r);
    r = is_prime_number(16);
    printf("%d\n", r);
    r = is_prime_number(17);
    printf("%d\n", r);
    r = is_prime_number(25);
    printf("%d\n", r);
    r = is_prime_number(-1);
    printf("%d\n", r);
    r = is_prime_number(113);
    printf("%d\n", r);
    r = is_prime_number(7919);
    printf("%d\n", r);
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 6-main.c 6-is_prime_number.c -o 6-prime
julien@ubuntu:~/0x08. Recursion$ ./6-prime
0
0
0
1
0
0
1
1
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `6-is_prime_number.c`

Done! Help Check your code Get a sandbox QA Review

7. They say we only use a fraction of our brain's true potential. Now that's when we're awake. When we're asleep, we can do almost anything

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that returns `1` if a string is a palindrome and `0` if not.

- Prototype: `int is_palindrome(char *s);`
- An empty string is a palindrome

```
julien@ubuntu:~/0x08. Recursion$ cat 100-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int r;

    r = is_palindrome("level");
    printf("%d\n", r);
    r = is_palindrome("redder");
    printf("%d\n", r);
    r = is_palindrome("test");
    printf("%d\n", r);
    r = is_palindrome("step on no pets");
    printf("%d\n", r);
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=g
nu89 100-main.c 100-is_palindrome.c -o 100-palindrome
julien@ubuntu:~/0x08. Recursion$ ./100-palindrome
1
1
0
```

```
1

julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `100-is_palindrome.c`

Done! Help Check your code Get a sandbox QA Review

8. Inception. Now, before you bother telling me it's impossible...

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that compares two strings and returns `1` if the strings can be considered identical, otherwise return `0`.

- Prototype: `int wildcmp(char *s1, char *s2);`
- `s2` can contain the special character `*`.
- The special char `*` can replace any string (including an empty string)

```
julien@ubuntu:~/0x08. Recursion$ cat 101-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int r;

    r = wildcmp("main.c", "*.c");
    printf("%d\n", r);
    r = wildcmp("main.c", "m*a*i*n*.*c*");
    printf("%d\n", r);
    r = wildcmp("main.c", "main.c");
    printf("%d\n", r);
    r = wildcmp("main.c", "m*c");
    printf("%d\n", r);
    r = wildcmp("main.c", "ma********************************c");
    printf("%d\n", r);
    r = wildcmp("main.c", "*");
    printf("%d\n", r);
    r = wildcmp("main.c", "***");
    printf("%d\n", r);
    r = wildcmp("main.c", "m.*c");
    printf("%d\n", r);
    r = wildcmp("main.c", "**.*c");
    printf("%d\n", r);
    r = wildcmp("main-main.c", "ma*in.c");
    printf("%d\n", r);
    r = wildcmp("main", "main*d");
    printf("%d\n", r);
    r = wildcmp("abc", "*b");
    printf("%d\n", r);
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=g
nu89 101-main.c 101-wildcmp.c -o 101-wildcmp
julien@ubuntu:~/0x08. Recursion$ ./101-wildcmp
1
1
1
1
1
1
1
0
1
1
0
```

```
0
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `101-wildcmp.c`

0x09. C - Static libraries

Concepts

For this project, we expect you to look at this concept:

- C static libraries

Resources

Read or watch:

- What Is A “C” Library? What Is It Good For?
- Creating A Static “C” Library Using “ar” and “ranlib”
- Using A “C” Library In A Program
- What is difference between Dynamic and Static library(Static and Dynamic linking) (*stop at 4:44*)

man or help:

- `ar`
- `ranlib`
- `nm`

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What is a static library, how does it work, how to create one, and how to use it
- Basic usage of `ar`, `ranlib`, `nm`

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

C

- Allowed editors: `vi`, `vim`, `emacs`

- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, etc… is forbidden
- You are allowed to use _putchar
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don't forget to push your header file

Bash

- Allowed editors: `vi`, `vim`, `emacs`
- All your scripts will be tested on Ubuntu 20.04 LTS
- All your files should end with a new line (why?)
- The first line of all your files should be exactly `#!/bin/bash`
- A `README.md` file, at the root of the folder of the project, describing what each script is doing
- All your files must be executable

More Info

You do not need to learn about dynamic libraries, yet.

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. A library is not a luxury but one of the necessities of life

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Create the static library `libmy.a` containing all the functions listed below:

```
int _putchar(char c);

int _islower(int c);

int _isalpha(int c);

int _abs(int n);
int _isupper(int c);
int _isdigit(int c);
int _strlen(char *s);
void _puts(char *s);
char *_strcpy(char *dest, char *src);
int _atoi(char *s);
char *_strcat(char *dest, char *src);
char *_strncat(char *dest, char *src, int n);
char *_strncpy(char *dest, char *src, int n);
int _strcmp(char *s1, char *s2);
char *_memset(char *s, char b, unsigned int n);
char *_memcpy(char *dest, char *src, unsigned int n);
char *_strchr(char *s, char c);
unsigned int _strspn(char *s, char *accept);
char *_strpbrk(char *s, char *accept);
char *_strstr(char *haystack, char *needle);
```

If you haven't coded all of the above functions create empty ones with the right prototype. Don't forget to push your `main.h` file to your repository. It should at least contain all the prototypes of the above functions.

```
julien@ubuntu:~/0x09. Static Librairies$ ar -t libmy.a
0-isupper.o
0-memset.o
0-strcat.o
1-isdigit.o
1-memcpy.o
1-strncat.o
100-atoi.o
2-strchr.o
2-strlen.o
2-strncpy.o
3-islower.o
3-puts.o
3-strcmp.o
```

```
3-strspn.o
4-isalpha.o
4-strpbrk.o
5-strstr.o
6-abs.o
9-strcpy.o
_putchar.o
julien@ubuntu:~/0x09. Static Librairies$ nm libmy.a

0-isupper.o:
0000000000000000 T _isupper

0-memset.o:
0000000000000000 T _memset

0-strcat.o:
0000000000000000 T _strcat

1-isdigit.o:
0000000000000000 T _isdigit

1-memcpy.o:
0000000000000000 T _memcpy

1-strncat.o:
0000000000000000 T _strncat

100-atoi.o:
0000000000000000 T _atoi

2-strchr.o:
0000000000000000 T _strchr
```

```
2-strlen.o:
0000000000000000 T _strlen

2-strncpy.o:
0000000000000000 T _strncpy

3-islower.o:
0000000000000000 T _islower

3-puts.o:
                 U _putchar
0000000000000000 T _puts

3-strcmp.o:
0000000000000000 T _strcmp

3-strspn.o:
0000000000000000 T _strspn

4-isalpha.o:
0000000000000000 T _isalpha

4-strpbrk.o:
0000000000000000 T _strpbrk

5-strstr.o:
0000000000000000 T _strstr

6-abs.o:
0000000000000000 T _abs

9-strcpy.o:
0000000000000000 T _strcpy
```

```
_putchar.o:
0000000000000000 T _putchar
                 U write
julien@ubuntu:~/0x09. Static Librairies$ cat main.c
#include "main.h"


int main(void)
{
    _puts("\"At the end of the day, my goal was to be the best hacker\"\n\t
- Kevin Mitnick");
    return (0);
}
julien@ubuntu:~/0x09. Static Librairies$ gcc -std=gnu89 main.c -L. -lmy -o
quote
julien@ubuntu:~/0x09. Static Librairies$ ./quote
"At the end of the day, my goal was to be the best hacker"
    - Kevin Mitnick
julien@ubuntu:~/0x09. Static Librairies$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x09-static_libraries`
- File: `libmy.a, main.h`

Done! Help Check your code QA Review

1. Without libraries what have we? We have no past and no future

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Create a script called `create_static_lib.sh` that creates a static library called `liball.a` from all the `.c` files that are in the current directory.

```
julien@ubuntu:~/0x09. Static Librairies$ ls *.c
0-isupper.c  0-strcat.c  1-isdigit.c  1-strncat.c  2-strlen.c   3-islower.c
3-strcmp.c  4-isalpha.c  5-strstr.c  9-strcpy.c  _putchar.c
0-memset.c   100-atoi.c  1-memcpy.c   2-strchr.c   2-strncpy.c  3-puts.c
3-strspn.c  4-strpbrk.c  6-abs.c
julien@ubuntu:~/0x09. Static Librairies$ ./create_static_lib.sh
```

```
julien@ubuntu:~/0x09. Static Librairies$ ls *.a
liball.a
julien@ubuntu:~/0x09. Static Librairies$ ar -t liball.a
0-isupper.o
0-memset.o
0-strcat.o
100-atoi.o
1-isdigit.o
1-memcpy.o
1-strncat.o
2-strchr.o
2-strlen.o
2-strncpy.o
3-islower.o
3-puts.o
3-strcmp.o
3-strspn.o
4-isalpha.o
4-strpbrk.o
5-strstr.o
6-abs.o
9-strcpy.o
_putchar.o
julien@ubuntu:~/0x09. Static Librairies$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x09-static_libraries`
- File: `create_static_lib.sh`

0x0A. C - argc, argv

Resources

Read or watch:

- Arguments to main
- argc and argv
- What does argc and argv mean?
- how to compile with unused variables

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- How to use arguments passed to your program
- What are two prototypes of `main` that you know of, and in which case do you use one or the other
- How to use `__attribute__((unused))` or `(void)` to compile functions with unused variables or parameters

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`

- Don’t forget to push your header file
- You are allowed to use the standard library

Quiz questions
Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. It ain't what they call you, it's what you answer to
mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that prints its name, followed by a new line.

- If you rename the program, it will print the new name, without having to compile it again
- You should not remove the path before the name of the program

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-whatsmyname.c -o mynameis
julien@ubuntu:~/0x0A. argc, argv$ ./mynameis
./mynameis
julien@ubuntu:~/0x0A. argc, argv$ mv mynameis mynewnameis
julien@ubuntu:~/0x0A. argc, argv$ ./mynewnameis
./mynewnameis
julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `0-whatsmyname.c`

Done! Help Check your code Get a sandbox QA Review

1. Silence is argument carried out by other means
mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that prints the number of arguments passed into it.

- Your program should print a number, followed by a new line

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-args.c -o nargs
julien@ubuntu:~/0x0A. argc, argv$ ./nargs
0
```

```
julien@ubuntu:~/0x0A. argc, argv$ ./nargs hello
1
julien@ubuntu:~/0x0A. argc, argv$ ./nargs "hello, world"
1
julien@ubuntu:~/0x0A. argc, argv$ ./nargs hello, world
2
julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `1-args.c`

Done! Help Check your code Get a sandbox QA Review

2. The best argument against democracy is a five-minute conversation with the average voter

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that prints all arguments it receives.

- All arguments should be printed, including the first one
- Only print one argument per line, ending with a new line

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-args.c -o args
julien@ubuntu:~/0x0A. argc, argv$ ./args
./args
julien@ubuntu:~/0x0A. argc, argv$ ./args You can do anything, but not everything.
./args
You
can
do
anything,
but
not
everything.
julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `2-args.c`

Done! Help Check your code Get a sandbox QA Review

3. Neither irony nor sarcasm is argument

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that multiplies two numbers.

- Your program should print the result of the multiplication, followed by a new line
- You can assume that the two numbers and result of the multiplication can be stored in an integer
- If the program does not receive two arguments, your program should print `Error`, followed by a new line, and return `1`

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-mul.c -o mul
julien@ubuntu:~/0x0A. argc, argv$ ./mul 2 3
6
julien@ubuntu:~/0x0A. argc, argv$ ./mul 2 -3
-6
julien@ubuntu:~/0x0A. argc, argv$ ./mul 2 0
0
julien@ubuntu:~/0x0A. argc, argv$ ./mul 245 3245342
795108790
julien@ubuntu:~/0x0A. argc, argv$ ./mul
Error
julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `3-mul.c`

Done! Help Check your code Get a sandbox QA Review

4. To infinity and beyond

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that adds positive numbers.

- Print the result, followed by a new line
- If no number is passed to the program, print `0`, followed by a new line
- If one of the number contains symbols that are not digits, print `Error`, followed by a new line, and return `1`
- You can assume that numbers and the addition of all the numbers can be stored in an `int`

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 4-add.c -o add

julien@ubuntu:~/0x0A. argc, argv$ ./add 1 1

2

julien@ubuntu:~/0x0A. argc, argv$ ./add 1 10 100 1000

1111

julien@ubuntu:~/0x0A. argc, argv$ ./add 1 2 3 e 4 5

Error

julien@ubuntu:~/0x0A. argc, argv$ ./add

0

julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `4-add.c`

Done! Help Check your code Get a sandbox QA Review

5. Minimal Number of Coins for Change

#advanced

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that prints the minimum number of coins to make change for an amount of money.

- Usage: `./change cents`
- where `cents` is the amount of cents you need to give back
- if the number of arguments passed to your program is not exactly `1`, print `Error`, followed by a new line, and return `1`
- you should use `atoi` to parse the parameter passed to your program
- If the number passed as the argument is negative, print `0`, followed by a new line
- You can use an unlimited number of coins of values 25, 10, 5, 2, and 1 cent

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 100-change.c -o change
```

```
julien@ubuntu:~/0x0A. argc, argv$ ./change

Error

julien@ubuntu:~/0x0A. argc, argv$ ./change 10

1

julien@ubuntu:~/0x0A. argc, argv$ ./change 100

4

julien@ubuntu:~/0x0A. argc, argv$ ./change 101

5

julien@ubuntu:~/0x0A. argc, argv$ ./change 13

3

julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `100-change.c`

0x0B. C - malloc, free

Resources

Read or watch:

- 0x0a - malloc & free - quick overview.pdf
- Dynamic memory allocation in C - malloc calloc realloc free (*stop at 6:50*)

man or help:

- `malloc`
- `free`

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What is the difference between automatic and dynamic allocation
- What is `malloc` and `free` and how to use them
- Why and when use `malloc`
- How to use `valgrind` to check for memory leak

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables

- No more than 5 functions per file
- The only C standard library functions allowed are `malloc` and `free`. Any use of functions like `printf`, `puts`, `calloc`, `realloc` etc… is forbidden
- You are allowed to use _putchar
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don't forget to push your header file

More Info

You do not have to learn about `calloc` and `realloc`.

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Float like a butterfly, sting like a bee

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that creates an array of chars, and initializes it with a specific char.

- Prototype: `char *create_array(unsigned int size, char c);`
- Returns `NULL` if size = `0`
- Returns a pointer to the array, or `NULL` if it fails

```
julien@ubuntu:~/0x0a. malloc, free$ cat 0-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
```

```
void simple_print_buffer(char *buffer, unsigned int size)
{
    unsigned int i;

    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", buffer[i]);
        i++;
    }
    printf("\n");
}

/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    char *buffer;

    buffer = create_array(98, 'H');
    if  (buffer == NULL)
    {
        printf("failed to allocate memory\n");
        return (1);
    }
    simple_print_buffer(buffer, 98);
    free(buffer);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-main.c 0-create_array.c -o a
julien@ubuntu:~/0x0a. malloc, free$ ./a
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `0-create_array.c`

Done! Help Check your code Get a sandbox QA Review

1. The woman who has no imagination has no wings

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that returns a pointer to a newly allocated space in memory, which contains a copy of the string given as a parameter.

- Prototype: `char *_strdup(char *str);`
- The `_strdup()` function returns a pointer to a new string which is a duplicate of the string `str`. Memory for the new string is obtained with `malloc`, and can be freed with `free`.
- Returns `NULL` if str = NULL

- On success, the `_strdup` function returns a pointer to the duplicated string. It returns `NULL` if insufficient memory was available

FYI: The standard library provides a similar function: `strdup`. Run `man strdup` to learn more.

```
julien@ubuntu:~/0x0a. malloc, free$ cat 1-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    char *s;

    s = _strdup("ALX SE");
    if (s == NULL)
    {
        printf("failed to allocate memory\n");
        return (1);
    }
    printf("%s\n", s);
    free(s);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-main.c 1-strdup.c -o s
julien@ubuntu:~/0x0a. malloc, free$ ./s
ALX SE
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `1-strdup.c`

Done! Help Check your code Get a sandbox QA Review

2. He who is not courageous enough to take risks will accomplish nothing in life

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that concatenates two strings.

- Prototype: `char *str_concat(char *s1, char *s2);`
- The returned pointer should point to a newly allocated space in memory which contains the contents of `s1`, followed by the contents of `s2`, and null terminated
- if `NULL` is passed, treat it as an empty string
- The function should return `NULL` on failure

```
julien@ubuntu:~/0x0a. malloc, free$ cat 2-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    char *s;

    s = str_concat("Betty ", "Holberton");
    if (s == NULL)
    {
        printf("failed\n");
        return (1);
    }
    printf("%s\n", s);
    free(s);
    return (0);

}

julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 2-main.c 2-str_concat.c -o c

julien@ubuntu:~/c/curriculum_by_julien/holbertonschool-low_level_programmin
g/0x0a. malloc, free$ ./c | cat -e

Betty Holberton$

julien@ubuntu:~/c/curriculum_by_julien/holbertonschool-low_level_programmin
g/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `2-str_concat.c`

Done! Help Check your code Get a sandbox QA Review

3. If you even dream of beating me you'd better wake up and apologize

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that returns a pointer to a 2 dimensional array of integers.

- Prototype: `int **alloc_grid(int width, int height);`
- Each element of the grid should be initialized to `0`
- The function should return `NULL` on failure
- If `width` or `height` is `0` or negative, return `NULL`

```
julien@ubuntu:~/0x0a. malloc, free$ cat 3-main.c

#include "main.h"

#include <stdio.h>

#include <stdlib.h>


/**

 * print_grid - prints a grid of integers

 * @grid: the address of the two dimensional grid

 * @width: width of the grid

 * @height: height of the grid

 *

 * Return: Nothing.

 */
```

```
void print_grid(int **grid, int width, int height)
{
    int w;
    int h;

    h = 0;
    while (h < height)
    {
        w = 0;
        while (w < width)
        {
            printf("%d ", grid[h][w]);
            w++;
        }
        printf("\n");
        h++;
    }
}

/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    int **grid;

    grid = alloc_grid(6, 4);
    if (grid == NULL)
    {
        return (1);
    }
    print_grid(grid, 6, 4);

    printf("\n");

    grid[0][3] = 98;

    grid[3][4] = 402;

    print_grid(grid, 6, 4);

    return (0);

}

julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 3-main.c 3-alloc_grid.c -o g

julien@ubuntu:~/0x0a. malloc, free$ ./g

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0


0 0 0 98 0 0

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 402 0

julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `3-alloc_grid.c`

Done! Help Check your code Get a sandbox QA Review

4. It's not bragging if you can back it up

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that frees a 2 dimensional grid previously created by your `alloc_grid` function.

- Prototype: `void free_grid(int **grid, int height);`
- Note that we will compile with your `alloc_grid.c` file. Make sure it compiles.

```
julien@ubuntu:~/0x0a. malloc, free$ cat 4-main.c

#include "main.h"
```

```
#include <stdio.h>
#include <stdlib.h>

/**
 * print_grid - prints a grid of integers
 * @grid: the address of the two dimensional grid
 * @width: width of the grid
 * @height: height of the grid
 *
 * Return: Nothing.
 */
void print_grid(int **grid, int width, int height)
{
    int w;
    int h;

    h = 0;
    while (h < height)
    {
        w = 0;
        while (w < width)
        {
            printf("%d ", grid[h][w]);
            w++;
        }
        printf("\n");
        h++;
    }
}

/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    int **grid;

    grid = alloc_grid(6, 4);
    if (grid == NULL)
    {
        return (1);
    }
    print_grid(grid, 6, 4);
    printf("\n");
    grid[0][3] = 98;
    grid[3][4] = 402;
    print_grid(grid, 6, 4);
    free_grid(grid, 4);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 4-main.c 3-alloc_grid.c 4-free_grid.c -o f
julien@ubuntu:~/0x0a. malloc, free$ valgrind ./f
==5013== Memcheck, a memory error detector
==5013== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==5013== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==5013== Command: ./f
==5013==
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 98 0 0
0 0 0 0 0 0
```

```
0 0 0 0 0 0

0 0 0 0 402 0

==5013==

==5013== HEAP SUMMARY:

==5013==     in use at exit: 0 bytes in 0 blocks

==5013==   total heap usage: 6 allocs, 6 frees, 1,248 bytes allocated

==5013==

==5013== All heap blocks were freed -- no leaks are possible

==5013==

==5013== For counts of detected and suppressed errors, rerun with: -v

==5013== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `4-free_grid.c`

Done! Help Check your code Get a sandbox QA Review

5. It isn't the mountains ahead to climb that wear you out; it's the pebble in your shoe

#advanced

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that concatenates all the arguments of your program.

- Prototype: `char *argstostr(int ac, char **av);`
- Returns `NULL` if `ac == 0` or `av == NULL`
- Returns a pointer to a new string, or `NULL` if it fails
- Each argument should be followed by a `\n` in the new string

```
julien@ubuntu:~/0x0a. malloc, free$ cat 100-main.c

#include "main.h"

#include <stdio.h>

#include <stdlib.h>


/**

 * main - check the code for ALX School students.

 *

 * Return: Always 0.
 */
int main(int ac, char *av[])
{
    char *s;

    s = argstostr(ac, av);
    if (s == NULL)
    {
        return (1);
    }
    printf("%s", s);
    free(s);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 100-main.c 100-argstostr.c -o args
julien@ubuntu:~/0x0a. malloc, free$ ./args I will "show you" how great I am
./args
I
will
show you
how
great
I
am
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `100-argstostr.c`

Done! Help Check your code Get a sandbox QA Review

6. I will show you how great I am

#advanced

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that splits a string into words.

- Prototype: `char **strtow(char *str);`
- The function returns a pointer to an array of strings (words)
- Each element of this array should contain a single word, null-terminated
- The last element of the returned array should be `NULL`
- Words are separated by spaces
- Returns `NULL` if `str == NULL` or `str == ""`
- If your function fails, it should return `NULL`

```
julien@ubuntu:~/0x0a. malloc, free$ cat 101-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * print_tab - Prints an array of string
 * @tab: The array to print
 *
 * Return: nothing
 */
void print_tab(char **tab)
{
    int i;

    for (i = 0; tab[i] != NULL; ++i)
    {
        printf("%s\n", tab[i]);
    }
}

/**
 * main - check the code for ALX School students.
 *
 * Return: 1 if an error occurred, 0 otherwise
 */
int main(void)
```

```
{
    char **tab;

    tab = strtow("      ALX School         #cisfun      ");
    if (tab == NULL)
    {
        printf("Failed\n");
        return (1);
    }
    print_tab(tab);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 101-main.c 101-strtow.c -o strtow
julien@ubuntu:~/0x0a. malloc, free$ ./strtow | cat -e
ALX$
School$
#cisfun$
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `101-strtow.c`

0x0B. C - malloc, free

Concepts

For this project, we expect you to look at this concept:

- Automatic and dynamic allocation, malloc and free

Resources

Read or watch:

- 0x0a - malloc & free - quick overview.pdf
- Dynamic memory allocation in C - malloc calloc realloc free (*stop at 6:50*)

man or help:

- `malloc`
- `free`

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What is the difference between automatic and dynamic allocation
- What is `malloc` and `free` and how to use them
- Why and when use `malloc`
- How to use `valgrind` to check for memory leak

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line

- A README.md file, at the root of the folder of the project is mandatory
- Your code should use the Betty style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- The only C standard library functions allowed are malloc and free. Any use of functions like printf, puts, calloc, realloc etc… is forbidden
- You are allowed to use _putchar
- You don't have to push _putchar.c, we will use our file. If you do it won't be taken into account
- In the following examples, the main.c files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own main.c files at compilation. Our main.c files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function _putchar should be included in your header file called main.h
- Don't forget to push your header file

More Info

You do not have to learn about calloc and realloc.

Quiz questions
Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Float like a butterfly, sting like a bee

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that creates an array of chars, and initializes it with a specific char.

- Prototype: `char *create_array(unsigned int size, char c);`
- Returns NULL if size = 0
- Returns a pointer to the array, or NULL if it fails

```
julien@ubuntu:~/0x0a. malloc, free$ cat 0-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(char *buffer, unsigned int size)
{
    unsigned int i;

    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", buffer[i]);
        i++;
    }
    printf("\n");
}

/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    char *buffer;

    buffer = create_array(98, 'H');
    if  (buffer == NULL)
    {
        printf("failed to allocate memory\n");
        return (1);
    }
    simple_print_buffer(buffer, 98);
    free(buffer);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 0-main.c 0-create_array.c -o a
julien@ubuntu:~/0x0a. malloc, free$ ./a
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
0x48 0x48 0x48 0x48 0x48 0x48 0x48 0x48
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `0-create_array.c`

Done! Help Check your code Get a sandbox QA Review

1. The woman who has no imagination has no wings

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that returns a pointer to a newly allocated space in memory, which contains a copy of the string given as a parameter.

- Prototype: `char *_strdup(char *str);`
- The `_strdup()` function returns a pointer to a new string which is a duplicate of the string `str`. Memory for the new string is obtained with `malloc`, and can be freed with `free`.
- Returns `NULL` if str = NULL
- On success, the `_strdup` function returns a pointer to the duplicated string. It returns `NULL` if insufficient memory was available

FYI: The standard library provides a similar function: `strdup`. Run `man strdup` to learn more.

```
julien@ubuntu:~/0x0a. malloc, free$ cat 1-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    char *s;

    s = _strdup("ALX SE");
    if (s == NULL)
    {
        printf("failed to allocate memory\n");
        return (1);
    }
    printf("%s\n", s);
    free(s);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 1-main.c 1-strdup.c -o s
julien@ubuntu:~/0x0a. malloc, free$ ./s
```

```
ALX SE
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `1-strdup.c`

Done! Help Check your code Get a sandbox QA Review

2. He who is not courageous enough to take risks will accomplish nothing in life

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that concatenates two strings.

- Prototype: `char *str_concat(char *s1, char *s2);`
- The returned pointer should point to a newly allocated space in memory which contains the contents of `s1`, followed by the contents of `s2`, and null terminated
- if `NULL` is passed, treat it as an empty string
- The function should return `NULL` on failure

```
julien@ubuntu:~/0x0a. malloc, free$ cat 2-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    char *s;

    s = str_concat("Betty ", "Holberton");
    if (s == NULL)
    {
        printf("failed\n");
        return (1);
    }
    printf("%s\n", s);
    free(s);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 2-main.c 2-str_concat.c -o c
julien@ubuntu:~/c/curriculum_by_julien/holbertonschool-low_level_programmin
g/0x0a. malloc, free$ ./c | cat -e
Betty Holberton$
julien@ubuntu:~/c/curriculum_by_julien/holbertonschool-low_level_programmin
g/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `2-str_concat.c`

Done! Help Check your code Get a sandbox QA Review

3. If you even dream of beating me you'd better wake up and apologize

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that returns a pointer to a 2 dimensional array of integers.

- Prototype: `int **alloc_grid(int width, int height);`
- Each element of the grid should be initialized to `0`
- The function should return `NULL` on failure
- If `width` or `height` is `0` or negative, return `NULL`

```
julien@ubuntu:~/0x0a. malloc, free$ cat 3-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * print_grid - prints a grid of integers
 * @grid: the address of the two dimensional grid
 * @width: width of the grid
 * @height: height of the grid
 *
 * Return: Nothing.
 */
void print_grid(int **grid, int width, int height)
{
    int w;
    int h;

    h = 0;
    while (h < height)
    {
        w = 0;
        while (w < width)
        {
            printf("%d ", grid[h][w]);
            w++;
        }
        printf("\n");
        h++;
    }
}

/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    int **grid;

    grid = alloc_grid(6, 4);
    if (grid == NULL)
    {
        return (1);
    }
    print_grid(grid, 6, 4);
    printf("\n");
    grid[0][3] = 98;
    grid[3][4] = 402;
    print_grid(grid, 6, 4);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 3-main.c 3-alloc_grid.c -o g
julien@ubuntu:~/0x0a. malloc, free$ ./g
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 98 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 402 0
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `3-alloc_grid.c`

Done! Help Check your code Get a sandbox QA Review

4. It's not bragging if you can back it up

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that frees a 2 dimensional grid previously created by your `alloc_grid` function.

- Prototype: `void free_grid(int **grid, int height);`

- Note that we will compile with your `alloc_grid.c` file. Make sure it compiles.

```
julien@ubuntu:~/0x0a. malloc, free$ cat 4-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * print_grid - prints a grid of integers
 * @grid: the address of the two dimensional grid
 * @width: width of the grid
 * @height: height of the grid
 *
 * Return: Nothing.
 */
void print_grid(int **grid, int width, int height)
{
    int w;
    int h;

    h = 0;
    while (h < height)
    {
        w = 0;
        while (w < width)
        {
            printf("%d ", grid[h][w]);
            w++;
        }
        printf("\n");
        h++;
    }
}
```

```
/**
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(void)
{
    int **grid;

    grid = alloc_grid(6, 4);
    if (grid == NULL)
    {
        return (1);
    }
    print_grid(grid, 6, 4);
    printf("\n");
    grid[0][3] = 98;
    grid[3][4] = 402;
    print_grid(grid, 6, 4);
    free_grid(grid, 4);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 4-main.c 3-alloc_grid.c 4-free_grid.c -o f
julien@ubuntu:~/0x0a. malloc, free$ valgrind ./f
==5013== Memcheck, a memory error detector
==5013== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==5013== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==5013== Command: ./f
==5013==
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

```
0 0 0 98 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 402 0
==5013==
==5013== HEAP SUMMARY:
==5013==     in use at exit: 0 bytes in 0 blocks
==5013==   total heap usage: 6 allocs, 6 frees, 1,248 bytes allocated
==5013==
==5013== All heap blocks were freed -- no leaks are possible
==5013==
==5013== For counts of detected and suppressed errors, rerun with: -v
==5013== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `4-free_grid.c`

Done! Help Check your code Get a sandbox QA Review

5. It isn't the mountains ahead to climb that wear you out; it's the pebble in your shoe

#advanced

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that concatenates all the arguments of your program.

- Prototype: `char *argstostr(int ac, char **av);`
- Returns `NULL` if `ac == 0` or `av == NULL`
- Returns a pointer to a new string, or `NULL` if it fails
- Each argument should be followed by a `\n` in the new string

```
julien@ubuntu:~/0x0a. malloc, free$ cat 100-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
```

```
 * main - check the code for ALX School students.
 *
 * Return: Always 0.
 */
int main(int ac, char *av[])
{
    char *s;

    s = argstostr(ac, av);
    if (s == NULL)
    {
        return (1);
    }
    printf("%s", s);
    free(s);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 100-main.c 100-argstostr.c -o args
julien@ubuntu:~/0x0a. malloc, free$ ./args I will "show you" how great I am
./args
I
will
show you
how
great
I
am
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `100-argstostr.c`

Done! Help Check your code Get a sandbox QA Review

6. I will show you how great I am

#advanced

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that splits a string into words.

- Prototype: `char **strtow(char *str);`
- The function returns a pointer to an array of strings (words)
- Each element of this array should contain a single word, null-terminated
- The last element of the returned array should be `NULL`
- Words are separated by spaces
- Returns `NULL` if `str == NULL` or `str == ""`
- If your function fails, it should return `NULL`

```
julien@ubuntu:~/0x0a. malloc, free$ cat 101-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * print_tab - Prints an array of string
 * @tab: The array to print
 *
 * Return: nothing
 */
void print_tab(char **tab)
{
    int i;

    for (i = 0; tab[i] != NULL; ++i)
    {
        printf("%s\n", tab[i]);
    }
}

/**
 * main - check the code for ALX School students.
 *
 * Return: 1 if an error occurred, 0 otherwise
 */
int main(void)
{
    char **tab;

    tab = strtow("      ALX School         #cisfun      ");
    if (tab == NULL)
    {
        printf("Failed\n");
        return (1);
    }
    print_tab(tab);
    return (0);
}
julien@ubuntu:~/0x0a. malloc, free$ gcc -Wall -pedantic -Werror -Wextra -st
d=gnu89 101-main.c 101-strtow.c -o strtow
julien@ubuntu:~/0x0a. malloc, free$ ./strtow | cat -e
ALX$
School$
#cisfun$
julien@ubuntu:~/0x0a. malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0B-malloc_free`
- File: `101-strtow.c`

0x0C. C - More malloc, free

CMemory allocation

Concepts

For this project, we expect you to look at this concept:

- Automatic and dynamic allocation, malloc and free

Resources

Read or watch:

- Do I cast the result of malloc?

man or help:

- `exit (3)`
- `calloc`
- `realloc`

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- How to use the `exit` function
- What are the functions `calloc` and `realloc` from the standard library and how to use them

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`

- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- The only C standard library functions allowed are `malloc`, `free` and `exit`. Any use of functions like `printf`, `puts`, `calloc`, `realloc` etc… is forbidden
- You are allowed to use _putchar
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don't forget to push your header file

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Trust no one

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that allocates memory using `malloc`.

- Prototype: `void *malloc_checked(unsigned int b);`
- Returns a pointer to the allocated memory
- if `malloc` fails, the `malloc_checked` function should cause normal process termination with a status value of `98`

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 0-main.c

#include "main.h"

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>


/**

 * main - check the code

 *

 * Return: Always 0.

 */
```

```
int main(void)
{
    char *c;
    int *i;
    float *f;
    double *d;

    c = malloc_checked(sizeof(char) * 1024);
    printf("%p\n", (void *)c);
    i = malloc_checked(sizeof(int) * 402);
    printf("%p\n", (void *)i);
    f = malloc_checked(sizeof(float) * 100000000);
    printf("%p\n", (void *)f);
    d = malloc_checked(INT_MAX);
    printf("%p\n", (void *)d);
    free(c);
    free(i);
    free(f);
    free(d);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 0-main.c 0-malloc_checked.c -o a
julien@ubuntu:~/0x0b. more malloc, free$ ./a
0x1e39010
0x1e39830
0x7f31f6c19010
julien@ubuntu:~/0x0b. more malloc, free$ echo $?
98
julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `0-malloc_checked.c`

Done! Help Check your code QA Review

1. string_nconcat

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that concatenates two strings.

- Prototype: `char *string_nconcat(char *s1, char *s2, unsigned int n);`
- The returned pointer shall point to a newly allocated space in memory, which contains `s1`, followed by the first `n` bytes of `s2`, and null terminated
- If the function fails, it should return `NULL`
- If `n` is greater or equal to the length of `s2` then use the entire string `s2`
- if `NULL` is passed, treat it as an empty string

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 1-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *concat;

    concat = string_nconcat("Best ", "School !!!", 6);
    printf("%s\n", concat);
    free(concat);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 1-main.c 1-string_nconcat.c -o 1-string_nconcat
julien@ubuntu:~/0x0b. more malloc, free$ ./1-string_nconcat
Best School
julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `1-string_nconcat.c`

Done! Help Check your code QA Review

2. _calloc

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that allocates memory for an array, using `malloc`.

- Prototype: `void *_calloc(unsigned int nmemb, unsigned int size);`
- The `_calloc` function allocates memory for an array of `nmemb` elements of `size` bytes each and returns a pointer to the allocated memory.
- The memory is set to zero
- If `nmemb` or `size` is `0`, then `_calloc` returns `NULL`
- If `malloc` fails, then `_calloc` returns `NULL`

FYI: The standard library provides a different function: `calloc`. Run `man calloc` to learn more.

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 2-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(char *buffer, unsigned int size)
{
    unsigned int i;

    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", buffer[i]);
        i++;
    }
    printf("\n");
}

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *a;

    a = _calloc(98, sizeof(char));
    strcpy(a, "Best");
    strcpy(a + 4, " School! :)\n");
    a[97] = '!';
    simple_print_buffer(a, 98);
    free(a);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 2-main.c 2-calloc.c -o 2-calloc
```

```
julien@ubuntu:~/0x0b. more malloc, free$ ./2-calloc
0x42 0x65 0x73 0x74 0x20 0x53 0x63 0x68 0x6f 0x6f
0x6c 0x21 0x20 0x3a 0x29 0x0a 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x21
julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `2-calloc.c`

Done! Help Check your code QA Review

3. array_range

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that creates an array of integers.

- Prototype: `int *array_range(int min, int max);`
- The array created should contain all the values from `min` (included) to `max` (included), ordered from `min` to `max`
- Return: the pointer to the newly created array
- If `min` > `max`, return `NULL`
- If `malloc` fails, return `NULL`

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 3-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(int *buffer, unsigned int size)
{
    unsigned int i;

    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", buffer[i]);
        i++;
    }
    printf("\n");
}

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int *a;


    a = array_range(0, 10);
    simple_print_buffer(a, 11);
    free(a);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 3-main.c 3-array_range.c -o 3-array_range
julien@ubuntu:~/0x0b. more malloc, free$ ./3-array_range
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
0x0a
julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `3-array_range.c`

Done! Help Check your code QA Review

4. _realloc

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that reallocates a memory block using `malloc` and `free`

- Prototype: `void *_realloc(void *ptr, unsigned int old_size, unsigned int new_size);`
- where `ptr` is a pointer to the memory previously allocated with a call to `malloc`: `malloc(old_size)`
- `old_size` is the size, in bytes, of the allocated space for `ptr`
- and `new_size` is the new size, in bytes of the new memory block
- The contents will be copied to the newly allocated space, in the range from the start of `ptr` up to the minimum of the old and new sizes
- If `new_size` > `old_size`, the “added” memory should not be initialized
- If `new_size` == `old_size` do not do anything and return `ptr`
- If `ptr` is `NULL`, then the call is equivalent to `malloc(new_size)`, for all values of `old_size` and `new_size`
- If `new_size` is equal to zero, and `ptr` is not `NULL`, then the call is equivalent to `free(ptr)`. Return `NULL`
- Don’t forget to free `ptr` when it makes sense

FYI: The standard library provides a different function: `realloc`. Run `man realloc` to learn more.

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 100-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(char *buffer, unsigned int size)
{
    unsigned int i;

    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", buffer[i]);
        i++;
    }
    printf("\n");
}
```

```
/**
 * main - check the code for
 *
 * Return: Always 0.
 */
int main(void)
{
    char *p;
    int i;

    p = malloc(sizeof(char) * 10);
    p = _realloc(p, sizeof(char) * 10, sizeof(char) * 98);
    i = 0;
    while (i < 98)
    {
        p[i++] = 98;
    }
    simple_print_buffer(p, 98);
    free(p);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 100-main.c 100-realloc.c -o 100-realloc
julien@ubuntu:~/0x0b. more malloc, free$ ./100-realloc
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
```

```
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62

julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `100-realloc.c`

Done! Help Check your code QA Review

5. We must accept finite disappointment, but never lose infinite hope

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that multiplies two positive numbers.

- Usage: `mul num1 num2`
- `num1` and `num2` will be passed in base 10
- Print the result, followed by a new line
- If the number of arguments is incorrect, print `Error`, followed by a new line, and exit with a status of `98`
- `num1` and `num2` should only be composed of digits. If not, print `Error`, followed by a new line, and exit with a status of `98`
- You are allowed to use more than 5 functions in your file

You can use `bc` (`man bc`) to check your results.

```
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 101-mul.c _putchar.c -o 101-mul

julien@ubuntu:~/0x0b. more malloc, free$ ./101-mul 10 98

980

julien@ubuntu:~/0x0b. more malloc, free$ ./101-mul 235234693269436436223446
526546334576437634765378653875874687649698659586695898579 28658034365084365
0834260831096791376082164086314308143086510846508164061340608316083108530861
037690137096750671305865708327607320967309780146073697395678645086340863048
074509730457034285809348250983420958324098503942850983425098342095834253452
674136392357558918799704645242261590747609149899354133505568757708070198930
692012471218551228363894170225521663160100130742587815831438704611827078935
778494086720405550894821603430854826123481453226898830252259887994523292902
811699275321605906519935117885185505475702845747159250069627382628886178404
353891403296687726447088

6741363923575589187997046452422615907476091498993541335055687577080701989306
9201247121855122836389417022552166316010013074258781583143870461182707893577
8494086720405550894821603430854826123481453226898830252259887994523292902811
6992753216059081057377926651337612618248332113256902485974371969385156015068
8138682740006839121878186016670586054186782843222372972136734824123929220681
5929149627431117020868905658535278284448472114084636774164996263864922950928
186789606720847417840215629497894071295951835184641385914179238085331
```

```
3812015295333546716634344284086426775480775747808150030732119704867805688704303461042373101473485092019906795014369069932

julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `101-mul.c`

0x0D. C - Preprocessor

Resources

Read or watch:

- Understanding C program Compilation Process
- Object-like Macros
- Macro Arguments
- Pre Processor Directives in C
- The C Preprocessor
- Standard Predefined Macros
- include guard
- Common Predefined Macros

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are macros and how to use them
- What are the most common predefined macros
- How to include guard your header files

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file

- The only C standard library functions allowed are `malloc`, `free` and `exit`. Any use of functions like `printf`, `puts`, `calloc`, `realloc` etc… is forbidden
- You are allowed to use _putchar
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- Don't forget to push your header file
- All your header files should be include guarded

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Object-like Macro

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Create a header file that defines a macro named `SIZE` as an abbreviation for the token `1024`.

```
julien@ubuntu:~/0x0c. macro, structures$ cat 0-main.c
#include "0-object_like_macro.h"
#include "0-object_like_macro.h"
#include <stdio.h>


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int s;

    s = 98 + SIZE;
    printf("%d\n", s);
    return (0);
}
```

```
julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 0-main.c -o a

julien@ubuntu:~/0x0c. macro, structures$ ./a

1122

julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `0-object_like_macro.h`

Done! Help Check your code QA Review

1. Pi

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Create a header file that defines a macro named `PI` as an abbreviation for the token `3.14159265359`.

```
julien@ubuntu:~/0x0c. macro, structures$ cat 1-main.c

#include "1-pi.h"

#include "1-pi.h"

#include <stdio.h>


/**

 * main - check the code

 *

 * Return: Always 0.

 */

int main(void)

{

    float a;

    float r;


    r = 98;

    a = PI * r * r;

    printf("%.3f\n", a);

    return (0);
}

julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 1-main.c -o b

julien@ubuntu:~/0x0c. macro, structures$ ./b

30171.855

julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `1-pi.h`

Done! Help Check your code QA Review

2. File name

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a program that prints the name of the file it was compiled from, followed by a new line.

- You are allowed to use the standard library

```
julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 2-main.c -o c

julien@ubuntu:~/0x0c. macro, structures$ ./c

2-main.c

julien@ubuntu:~/0x0c. macro, structures$ cp 2-main.c 02-main.c

julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 02-main.c -o cc

julien@ubuntu:~/0x0c. macro, structures$ ./cc

02-main.c

julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `2-main.c`

Done! Help Check your code Get a sandbox QA Review

3. Function-like macro

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function-like macro `ABS(x)` that computes the absolute value of a number `x`.

```
julien@ubuntu:~/0x0c. macro, structures$ cat 3-main.c
#include <stdio.h>
#include "3-function_like_macro.h"
#include "3-function_like_macro.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int i;
    int j;

    i = ABS(-98) * 10;
    j = ABS(98) * 10;
    printf("%d, %d\n", i, j);
    return (0);
}
julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 3-main.c -o d
julien@ubuntu:~/0x0c. macro, structures$ ./d
980, 980
julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `3-function_like_macro.h`

Done! Help Check your code QA Review

4. SUM

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function-like macro `SUM(x, y)` that computes the sum of the numbers `x` and `y`.

```
julien@ubuntu:~/0x0c. macro, structures$ cat 4-main.c
#include <stdio.h>
#include "4-sum.h"
#include "4-sum.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int s;

    s = SUM(98, 1024);
    printf("%d\n", s);
    return (0);
}
julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 4-main.c -o e
julien@ubuntu:~/0x0c. macro, structures$ ./e
1122
julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `4-sum.h`

0x0E. C - Structures, typedef

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are structures, when, why and how to use them
- How to use `typedef`

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- The only C standard library functions allowed are `printf`, `malloc`, `free` and `exit`.
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- Don't forget to push your header file
- All your header files should be include guarded

Quiz questions
Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Poppy

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Define a new type `struct dog` with the following elements:

- `name`, type = `char *`
- `age`, type = `float`
- `owner`, type = `char *`

```
julien@ubuntu:~/0x0d. structures, typedef$ cat 0-main.c
#include <stdio.h>
#include "dog.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    struct dog my_dog;

    my_dog.name = "Poppy";
    my_dog.age = 3.5;
    my_dog.owner = "Bob";
    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog.name, my_dog
.age);
    return (0);
}
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wex
tra -std=gnu89 0-main.c -o a
julien@ubuntu:~/0x0d. structures, typedef$ ./a
My name is Poppy, and I am 3.5 :) - Woof!
julien@ubuntu:~/0x0d. structures, typedef$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0E-structures_typedef`
- File: `dog.h`

Done! Help Check your code QA Review

1. A dog is the only thing on earth that loves you more than you love yourself

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that initialize a variable of type `struct dog`

- Prototype: `void init_dog(struct dog *d, char *name, float age, char *owner);`

```
julien@ubuntu:~/0x0d. structures, typedef$ cat 1-main.c
#include <stdio.h>
#include "dog.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    struct dog my_dog;


    init_dog(&my_dog, "Poppy", 3.5, "Bob");
    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog.name, my_dog
.age);
    return (0);
}
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wex
tra -std=gnu89 1-main.c 1-init_dog.c -o b
julien@ubuntu:~/0x0d. structures, typedef$ ./b
My name is Poppy, and I am 3.5 :) - Woof!
julien@ubuntu:~/0x0d. structures, typedef$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0E-structures_typedef`
- File: `1-init_dog.c`

Done! Help Check your code QA Review

2. A dog will teach you unconditional love. If you can have that in your life, things won't be too bad

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints a `struct dog`

- Prototype: `void print_dog(struct dog *d);`
- Format: see example bellow
- You are allowed to use the standard library
- If an element of `d` is `NULL`, print `(nil)` instead of this element. (if `name` is `NULL`, print `Name: (nil)`)
- If `d` is `NULL` print nothing.

```
julien@ubuntu:~/0x0d. structures, typedef$ cat 2-main.c
#include <stdio.h>
#include "dog.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    struct dog my_dog;

    my_dog.name = "Poppy";
    my_dog.age = 3.5;
    my_dog.owner = "Bob";
    print_dog(&my_dog);
    return (0);
}
```

```
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wex
tra -std=gnu89 2-main.c 2-print_dog.c -o c

julien@ubuntu:~/0x0d. structures, typedef$ ./c

Name: Poppy

Age: 3.500000

Owner: Bob

julien@ubuntu:~/0x0d. structures, typedef$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0E-structures_typedef`
- File: `2-print_dog.c`

Done! Help Check your code QA Review

3. Outside of a dog, a book is a man's best friend. Inside of a dog it's too dark to read

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Define a new type `dog_t` as a new name for the type `struct dog`.

```
julien@ubuntu:~/0x0d. structures, typedef$ cat 3-main.c

#include <stdio.h>

#include "dog.h"


/**

 * main - check the code

 *

 * Return: Always 0.

 */

int main(void)

{

    dog_t my_dog;


    my_dog.name = "Poppy";

    my_dog.age = 3.5;

    my_dog.owner = "Bob";
    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog.name, my_dog
.age);

    return (0);

}

julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wex
tra -std=gnu89 3-main.c -o d

julien@ubuntu:~/0x0d. structures, typedef$ ./d

My name is Poppy, and I am 3.5 :) - Woof!

julien@ubuntu:~/0x0d. structures, typedef$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0E-structures_typedef`
- File: `dog.h`

Done! Help Check your code QA Review

4. A door is what a dog is perpetually on the wrong side of

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that creates a new dog.

- Prototype: `dog_t *new_dog(char *name, float age, char *owner);`
- You have to store a copy of `name` and `owner`
- Return `NULL` if the function fails

```
julien@ubuntu:~/0x0d. structures, typedef$ cat 4-main.c

#include <stdio.h>

#include "dog.h"


/**

 * main - check the code

 *

 * Return: Always 0.

 */

int main(void)

{

    dog_t *my_dog;
    my_dog = new_dog("Poppy", 3.5, "Bob");
    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog->name, my_do
g->age);
    return (0);
}
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wex
tra -std=gnu89 4-main.c 4-new_dog.c -o e
julien@ubuntu:~/0x0d. structures, typedef$ ./e
My name is Poppy, and I am 3.5 :) - Woof!
julien@ubuntu:~/0x0d. structures, typedef$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0E-structures_typedef`
- File: `4-new_dog.c`

Done! Help Check your code QA Review

5. How many legs does a dog have if you call his tail a leg? Four. Saying that a tail is a leg doesn't make it a leg

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that frees dogs.

- Prototype: `void free_dog(dog_t *d);`

```
julien@ubuntu:~/0x0d. structures, typedef$ cat 5-main.c
#include <stdio.h>
#include "dog.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    dog_t *my_dog;


    my_dog = new_dog("Poppy", 3.5, "Bob");

    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog->name, my_do
g->age);

    free_dog(my_dog);

    return (0);

}

julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wex
tra -std=gnu89 5-main.c 5-free_dog.c 4-new_dog.c -o f

julien@ubuntu:~/0x0d. structures, typedef$ valgrind ./f

==22840== Memcheck, a memory error detector

==22840== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.

==22840== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright inf
o

==22840== Command: ./f

==22840==

My name is Poppy, and I am 3.5 :) - Woof!

==22840==

==22840== HEAP SUMMARY:

==22840==     in use at exit: 0 bytes in 0 blocks

==22840==   total heap usage: 4 allocs, 4 frees, 1,059 bytes allocated

==22840==

==22840== All heap blocks were freed -- no leaks are possible

==22840==

==22840== For counts of detected and suppressed errors, rerun with: -v

==22840== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

julien@ubuntu:~/0x0d. structures, typedef$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0E-structures_typedef`
- File: `5-free_dog.c`

0x0F. C - Function pointers

Resources

Read or watch:

- Function Pointer in C
- Pointers to functions
- Function Pointers in C / C++
- why pointers to functions?
- Everything you need to know about pointers in C

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are function pointers and how to use them
- What does a function pointer exactly hold
- Where does a function pointer point to in the virtual memory

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- The only C standard library functions allowed are `malloc`, `free` and `exit`. Any use of functions like `printf`, `puts`, `calloc`, `realloc` etc… is forbidden
- You are allowed to use _putchar

- You don’t have to push `_putchar.c`, we will use our file. If you do it won’t be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don’t have to push them to your repo (if you do we won’t take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `function_pointers.h`
- Don’t forget to push your header file
- All your header files should be include guarded

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. What's my name

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Write a function that prints a name.

- Prototype: `void print_name(char *name, void (*f)(char *));`

```
julien@ubuntu:~/0x0e. Function pointers$ cat 0-main.c
#include <stdio.h>
#include "function_pointers.h"

/**
 * print_name_as_is - prints a name as is
 * @name: name of the person
 *
 * Return: Nothing.
 */
void print_name_as_is(char *name)
{
    printf("Hello, my name is %s\n", name);
}

/**
 * print_name_uppercase - print a name in uppercase
 * @name: name of the person
 *
 * Return: Nothing.
 */
void print_name_uppercase(char *name)
{
    unsigned int i;

    printf("Hello, my uppercase name is ");
    i = 0;
    while (name[i])
    {
        if (name[i] >= 'a' && name[i] <= 'z')
        {
            putchar(name[i] + 'A' - 'a');
        }
        else
        {
            putchar(name[i]);
        }
        i++;
    }
}

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_name("Bob", print_name_as_is);
    print_name("Bob Dylan", print_name_uppercase);
    printf("\n");
    return (0);
}
julien@ubuntu:~/0x0e. Function pointers$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 0-main.c 0-print_name.c -o a
julien@ubuntu:~/0x0e. Function pointers$ ./a
Hello, my name is Bob
Hello, my uppercase name is BOB DYLAN
julien@ubuntu:~/0x0e. Function pointers$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0F-function_pointers`
- File: `0-print_name.c`

Done! Help Check your code Get a sandbox QA Review

1. If you spend too much time thinking about a thing, you'll never get it done

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Write a function that executes a function given as a parameter on each element of an array.

- Prototype: `void array_iterator(int *array, size_t size, void (*action)(int));`
- where `size` is the size of the array
- and `action` is a pointer to the function you need to use

```
julien@ubuntu:~/0x0e. Function pointers$ cat 1-main.c
#include <stdio.h>
#include "function_pointers.h"


/**
 * print_elem - prints an integer
 * @elem: the integer to print
 *
 * Return: Nothing.
 */
void print_elem(int elem)
{
    printf("%d\n", elem);
}

/**
 * print_elem_hex - prints an integer, in hexadecimal
 * @elem: the integer to print
 *
 * Return: Nothing.
 */
void print_elem_hex(int elem)
{
    printf("0x%x\n", elem);
}

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int array[5] = {0, 98, 402, 1024, 4096};

    array_iterator(array, 5, &print_elem);
    array_iterator(array, 5, &print_elem_hex);
    return (0);
}
julien@ubuntu:~/0x0e. Function pointers$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 1-main.c 1-array_iterator.c -o b
julien@ubuntu:~/0x0e. Function pointers$ ./b
0
98
402
1024
```

```
4096
0x0
0x62
0x192
0x400
0x1000
julien@ubuntu:~//0x0e. Function pointers$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0F-function_pointers`
- File: `1-array_iterator.c`

Done! Help Check your code Get a sandbox QA Review

2. To hell with circumstances; I create opportunities

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Write a function that searches for an integer.

- Prototype: `int int_index(int *array, int size, int (*cmp)(int));`
- where `size` is the number of elements in the array `array`
- `cmp` is a pointer to the function to be used to compare values
- `int_index` returns the index of the first element for which the `cmp` function does not return `0`
- If no element matches, return `-1`
- If size <= `0`, return `-1`

```
julien@ubuntu:~/0x0e. Function pointers$ cat 2-main.c
#include <stdio.h>
#include "function_pointers.h"


/**
 * is_98 - check if a number is equal to 98
 * @elem: the integer to check
 *
 * Return: 0 if false, something else otherwise.
 */
int is_98(int elem)
{
    return (98 == elem);
}

/**
 * is_strictly_positive - check if a number is greater than 0
 * @elem: the integer to check
 *
 * Return: 0 if false, something else otherwise.
 */
int is_strictly_positive(int elem)
{
    return (elem > 0);
}



/**
 * abs_is_98 - check if the absolute value of a number is 98
 * @elem: the integer to check
 *
 * Return: 0 if false, something else otherwise.
 */
int abs_is_98(int elem)
{
    return (elem == 98 || -elem == 98);
}

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int array[20] = {0, -98, 98, 402, 1024, 4096, -1024, -98, 1, 2, 3, 4, 5
, 6, 7, 8, 9, 10, 11, 98};
    int index;


    index = int_index(array, 20, is_98);
    printf("%d\n", index);
    index = int_index(array, 20, abs_is_98);
    printf("%d\n", index);
    index = int_index(array, 20, is_strictly_positive);
    printf("%d\n", index);
    return (0);
}
julien@ubuntu:~/0x0e. Function pointers$ gcc -Wall -pedantic -Werror -Wextr
a -std=gnu89 2-main.c 2-int_index.c -o c
julien@ubuntu:~/0x0e. Function pointers$ ./c
2
1
2
julien@ubuntu:~/0x0e. Function pointers$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0F-function_pointers`
- File: `2-int_index.c`

Done! Help Check your code Get a sandbox QA Review

3. A goal is not always meant to be reached, it often serves simply as something to aim at

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Write a program that performs simple operations.

- You are allowed to use the standard library
- Usage: `calc num1 operator num2`
- You can assume `num1` and `num2` are integers, so use the `atoi` function to convert them from the string input to `int`
- `operator` is one of the following:
  - `+`: addition
  - `-`: subtraction
  - `*`: multiplication
  - `/`: division

    - `%`: modulo
- The program prints the result of the operation, followed by a new line
- You can assume that the result of all operations can be stored in an `int`
- if the number of arguments is wrong, print `Error`, followed by a new line, and exit with the status `98`
- if the `operator` is none of the above, print `Error`, followed by a new line, and exit with the status `99`
- if the user tries to divide (`/` or `%`) by `0`, print `Error`, followed by a new line, and exit with the status `100`

This task requires that you create four different files.

3-calc.h

This file should contain all the function prototypes and data structures used by the program. You can use this structure:

```
/**
 * struct op - Struct op
 *
 * @op: The operator
 * @f: The function associated
 */
typedef struct op
{
    char *op;
    int (*f)(int a, int b);
} op_t;
```

3-op_functions.c

This file should contain the 5 following functions (not more):

- `op_add`: returns the sum of `a` and `b`. Prototype: `int op_add(int a, int b);`
- `op_sub`: returns the difference of `a` and `b`. Prototype: `int op_sub(int a, int b);`
- `op_mul`: returns the product of `a` and `b`. Prototype: `int op_mul(int a, int b);`
- `op_div`: returns the result of the division of `a` by `b`. Prototype: `int op_div(int a, int b);`
- `op_mod`: returns the remainder of the division of `a` by `b`. Prototype: `int op_mod(int a, int b);`

3-get_op_func.c

This file should contain the function that selects the correct function to perform the operation asked by the user. You're not allowed to declare any other function.

- Prototype: `int (*get_op_func(char *s))(int, int);`
- where `s` is the operator passed as argument to the program

- This function returns a pointer to the function that corresponds to the operator given as a parameter. Example: `get_op_func("+")` should return a pointer to the function `op_add`
- You are not allowed to use `switch` statements
- You are not allowed to use `for` or `do ... while` loops
- You are not allowed to use `goto`
- You are not allowed to use `else`
- You are not allowed to use more than one `if` statement in your code
- You are not allowed to use more than one `while` loop in your code
- If `s` does not match any of the 5 expected operators (`+`, `-`, `*`, `/`, `%`), return `NULL`
- You are only allowed to declare these two variables in this function:

```
    op_t ops[] = {
        {"+", op_add},
        {"-", op_sub},
        {"*", op_mul},
        {"/", op_div},
        {"%", op_mod},
        {NULL, NULL}
    };
    int i;
```

3-main.c

This file should contain your `main` function only.

- You are not allowed to code any other function than `main` in this file
- You are not allowed to directly call `op_add`, `op_sub`, `op_mul`, `op_div` or `op_mod` from the `main` function
- You have to use `atoi` to convert arguments to `int`
- You are not allowed to use any kind of loop
- You are allowed to use a maximum of 3 `if` statements

Compilation and examples

```
julien@ubuntu:~/0x0e. Function pointers$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-main.c 3-op_functions.c 3-get_op_func.c -o calc
julien@ubuntu:~/0x0e. Function pointers$ ./calc 1 + 1
2
julien@ubuntu:~/0x0e. Function pointers$ ./calc 97 + 1
98
julien@ubuntu:~/0x0e. Function pointers$ ./calc 1024 / 10
102
julien@ubuntu:~/0x0e. Function pointers$ ./calc 1024 '*' 98
```

```
100352
julien@ubuntu:~/0x0e. Function pointers$ ./calc 1024 '\*' 98
Error
julien@ubuntu:~/0x0e. Function pointers$ ./calc 1024 - 98
926
julien@ubuntu:~/0x0e. Function pointers$ ./calc 1024 '%' 98
44
julien@ubuntu:~/0x0e. Function pointers$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0F-function_pointers`
- File: `3-main.c, 3-op_functions.c, 3-get_op_func.c, 3-calc.h`

Done! Help Check your code Get a sandbox QA Review

4. Most hackers are young because young people tend to be adaptable. As long as you remain adaptable, you can always be a good hacker

#advanced

Score: 0.0% (*Checks completed: 0.0%*)

Write a program that prints the opcodes of its own main function.

- Usage: `./main number_of_bytes`
- Output format:
  - the opcodes should be printed in hexadecimal, lowercase
  - each opcode is two char long
  - listing ends with a new line
  - see example
- You are allowed to use `printf` and `atoi`
- You have to use `atoi` to convert the argument to an `int`
- If the number of argument is not the correct one, print `Error`, followed by a new line, and exit with the status `1`
- If the number of bytes is negative, print `Error`, followed by a new line, and exit with the status `2`
- You do not have to compile with any flags

Note: if you want to translate your opcodes to assembly instructions, you can use, for instance udcli.

```
julien@ubuntu:~/0x0e. Function pointers$ gcc -std=gnu89 100-main_opcodes.c -o main
julien@ubuntu:~/0x0e. Function pointers$ ./main 21
55 48 89 e5 48 83 ec 30 89 7d dc 48 89 75 d0 83 7d dc 02 74 14
julien@ubuntu:~/0x0e. Function pointers$ objdump -d -j.text -M intel main
[...]
```

```
00000000004005f6 <main>:
  4005f6:   55                      push   rbp
  4005f7:   48 89 e5                mov    rbp,rsp
  4005fa:   48 83 ec 30             sub    rsp,0x30
  4005fe:   89 7d dc                mov    DWORD PTR [rbp-0x24],edi
  400601:   48 89 75 d0             mov    QWORD PTR [rbp-0x30],rsi
  400605:   83 7d dc 02             cmp    DWORD PTR [rbp-0x24],0x2
  400609:   74 14                   je     40061f <main+0x29>
[...]
julien@ubuntu:~/0x0e. Function pointers$ ./main 21 | udcli -64 -x -o 4005f6
00000000004005f6 55                push rbp
00000000004005f7 4889e5            mov rbp, rsp
00000000004005fa 4883ec30          sub rsp, 0x30
00000000004005fe 897ddc            mov [rbp-0x24], edi
0000000000400601 488975d0          mov [rbp-0x30], rsi
0000000000400605 837ddc02          cmp dword [rbp-0x24], 0x2
0000000000400609 7414              jz 0x40061f
julien@ubuntu:~/0x0e. Function pointers$
```

- *Note 0: `je` is equivalent to `jz`*
- *Note 1: depending on how you write your `main` function, and on which machine you compile your program, the opcodes (and by extension the assembly code) might be different than the above example*

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0F-function_pointers`
- File: `100-main_opcodes.c`

0x10. C - Variadic functions

C

- By: Julien Barbier
- Weight: 1
- Project will start Jan 19, 2023 6:00 AM, must end by Jan 20, 2023 6:00 AM
- will be released at Jan 19, 2023 12:00 PM
- An auto review will be launched at the deadline

Resources

Read or watch:

- stdarg.h
- Variadic Functions
- Const Keyword

man or help:

- `stdarg`

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What are variadic functions
- How to use `va_start`, `va_arg` and `va_end` macros
- Why and how to use the `const` type qualifier

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`

- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- The only C standard library functions allowed are `malloc`, `free` and `exit`. Any use of functions like `printf`, `puts`, `calloc`, `realloc` etc… is forbidden
- You are allowed to use the following macros: `va_start`, `va_arg` and `va_end`
- You are allowed to use _putchar
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `variadic_functions.h`
- Don't forget to push your header file
- All your header files should be include guarded

Tasks

0. Beauty is variable, ugliness is constant

mandatory

Write a function that returns the sum of all its parameters.

- Prototype: `int sum_them_all(const unsigned int n, ...);`
- If `n == 0`, return `0`

```
julien@ubuntu:~/0x0f. variadic functions$ cat 0-main.c
#include <stdio.h>
#include "variadic_functions.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int sum;

    sum = sum_them_all(2, 98, 1024);

    printf("%d\n", sum);

    sum = sum_them_all(4, 98, 1024, 402, -1024);

    printf("%d\n", sum);

    return (0);

}

julien@ubuntu:~/0x0f. variadic functions$ gcc -Wall -pedantic -Werror -Wext
ra -std=gnu89 0-main.c 0-sum_them_all.c -o a

julien@ubuntu:~/0x0f. variadic functions$ ./a

1122

500

julien@ubuntu:~/0x0f. variadic functions$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x10-variadic_functions`
- File: `0-sum_them_all.c`

Done? Help Get a sandbox

1. To be is to be the value of a variable

mandatory

Write a function that prints numbers, followed by a new line.

- Prototype: `void print_numbers(const char *separator, const unsigned int n, ...);`
- where `separator` is the string to be printed between numbers
- and `n` is the number of integers passed to the function
- You are allowed to use `printf`
- If `separator` is `NULL`, don’t print it
- Print a new line at the end of your function

```
julien@ubuntu:~/0x0f. variadic functions$ cat 1-main.c

#include "variadic_functions.h"



/**

 * main - check the code

 *

 * Return: Always 0.

 */

int main(void)
```

```
{
    print_numbers(", ", 4, 0, 98, -1024, 402);
    return (0);
}
julien@ubuntu:~/0x0f. variadic functions$ gcc -Wall -pedantic -Werror -Wext
ra -std=gnu89 1-main.c 1-print_numbers.c -o b
julien@ubuntu:~/0x0f. variadic functions$ ./b
0, 98, -1024, 402
julien@ubuntu:~/0x0f. variadic functions$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x10-variadic_functions`
- File: `1-print_numbers.c`

Done? Help Get a sandbox

2. One woman's constant is another woman's variable

mandatory

Write a function that prints strings, followed by a new line.

- Prototype: `void print_strings(const char *separator, const unsigned int n, ...);`
- where `separator` is the string to be printed between the strings
- and `n` is the number of strings passed to the function
- You are allowed to use `printf`
- If separator is NULL, don't print it
- If one of the string is NULL, print `(nil)` instead
- Print a new line at the end of your function

```
julien@ubuntu:~/0x0f. Variadic functions$ cat 2-main.c
#include "variadic_functions.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_strings(", ", 2, "Jay", "Django");
    return (0);
}
julien@ubuntu:~/0x0f. Variadic functions$ gcc -Wall -pedantic -Werror -Wext
ra -std=gnu89 2-main.c 2-print_strings.c -o c
julien@ubuntu:~/0x0f. Variadic functions$ ./c
Jay, Django
julien@ubuntu:~/0x0f. Variadic functions$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x10-variadic_functions`
- File: `2-print_strings.c`

Done? Help Get a sandbox

3. To be is a to be the value of a variable

mandatory

Write a function that prints anything.

- Prototype: `void print_all(const char * const format, ...);`
- where `format` is a list of types of arguments passed to the function
  - `c`: `char`
  - `i`: `integer`
  - `f`: `float`
  - `s`: `char *` (if the string is NULL, print `(nil)` instead
  - any other char should be ignored
  - see example
- You are not allowed to use `for`, `goto`, ternary operator, `else`, `do ... while`
- You can use a maximum of
  - 2 `while` loops
  - 2 `if`
- You can declare a maximum of `9` variables
- You are allowed to use `printf`
- Print a new line at the end of your function

```
julien@ubuntu:~/0x0f. Variadic functions$ cat 3-main.c
#include "variadic_functions.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
```

```
{
    print_all("ceis", 'B', 3, "stSchool");
    return (0);
}
julien@ubuntu:~/0x0f. Variadic functions$ gcc -Wall -pedantic -Werror -Wext
ra -std=gnu89 3-main.c 3-print_all.c -o d
julien@ubuntu:~/0x0f. Variadic functions$ ./d
B, 3, stSchool
julien@ubuntu:~/0x0f. Variadic functions$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x10-variadic_functions`
- File: `3-print_all.c`