At the end of this project, you are expected to be able to explain to anyone (/rltoken/-ERX2Jh115GIdTUsSixnnA), **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 (https://github.com/holbertonschool/Betty/blob/master/betty-style.pl) and betty-doc.pl (https://github.com/holbertonschool/Betty/blob/master/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 (https://github.com/holbertonschool/_putchar.c/blob/master/_putchar.c)
- 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! (Hide quiz)

Question #0

Which of these loop statements don't exist?

- [ ] `for`
- [ ] `while`
- [x] `foreach`
- [ ] `do... while`
- [x] `loop_to`
- [x] `each`

Question #1

What is the ASCII value of `A` ?

- ( ) 97
- (•) 65
- ( ) 12
- ( ) 1

Question #2

What is the ASCII value of `a` ?

- (•) 97
- ( ) 65
- ( ) 12
- ( ) 1

Question #3

(/)

What is the ASCII value of `J` ?

- 70
- 72
- 74
- 76

Question #4

What is the ASCII value of `0` ?

- 79
- 0
- 48

Question #5

What is the ASCII value of `-` ?

- 45
- 3
- 47

Question #6

What is the ASCII value of `5` ?

- 50
- 53
- 5

Question #7

What is the result of `12 % 2` ?

- 0
- 1
- 2

Question #8
(/)

What is the result of `12 % 3` ?

- [x] 0
- [ ] 1
- [ ] 2
- [ ] 3

Question #9

What is the result of `12 % 10` ?

- [ ] 0
- [ ] 1
- [x] 2
- [ ] 3

Question #10

What is the result of `89 % 7` ?

- [ ] 0
- [ ] 2
- [ ] 3
- [x] 5

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 _putchar.c 0-p
utchar.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 _putchar.c 1-ma
in.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 _putchar.c 2-ma
in.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 _putchar.c 3-ma
in.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 _putchar.c 4-ma
in.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 _putchar.c 5-ma
in.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`

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 _putchar.c 6-ma
in.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 _putchar.c 7-ma
in.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 (/rltoken/aNwRcWg7MPM1J2lYuuuBjA) 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 _putchar.c 8-ma
in.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 _putchar.c 9-ma
in.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

mandatory

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-m
ain.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 _putchar.c 11-m
ain.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, 21, 22, 2
3, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 4
4, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 6
5, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 8
6, 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, 1
3, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 3
4, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 5
5, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 7
6, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 9
7, 98
julien@ubuntu:~/0x02$
```

Repo:

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

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 _putchar.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,...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`

Done! | Help | Check your code | >_ Get a sandbox | QA Review