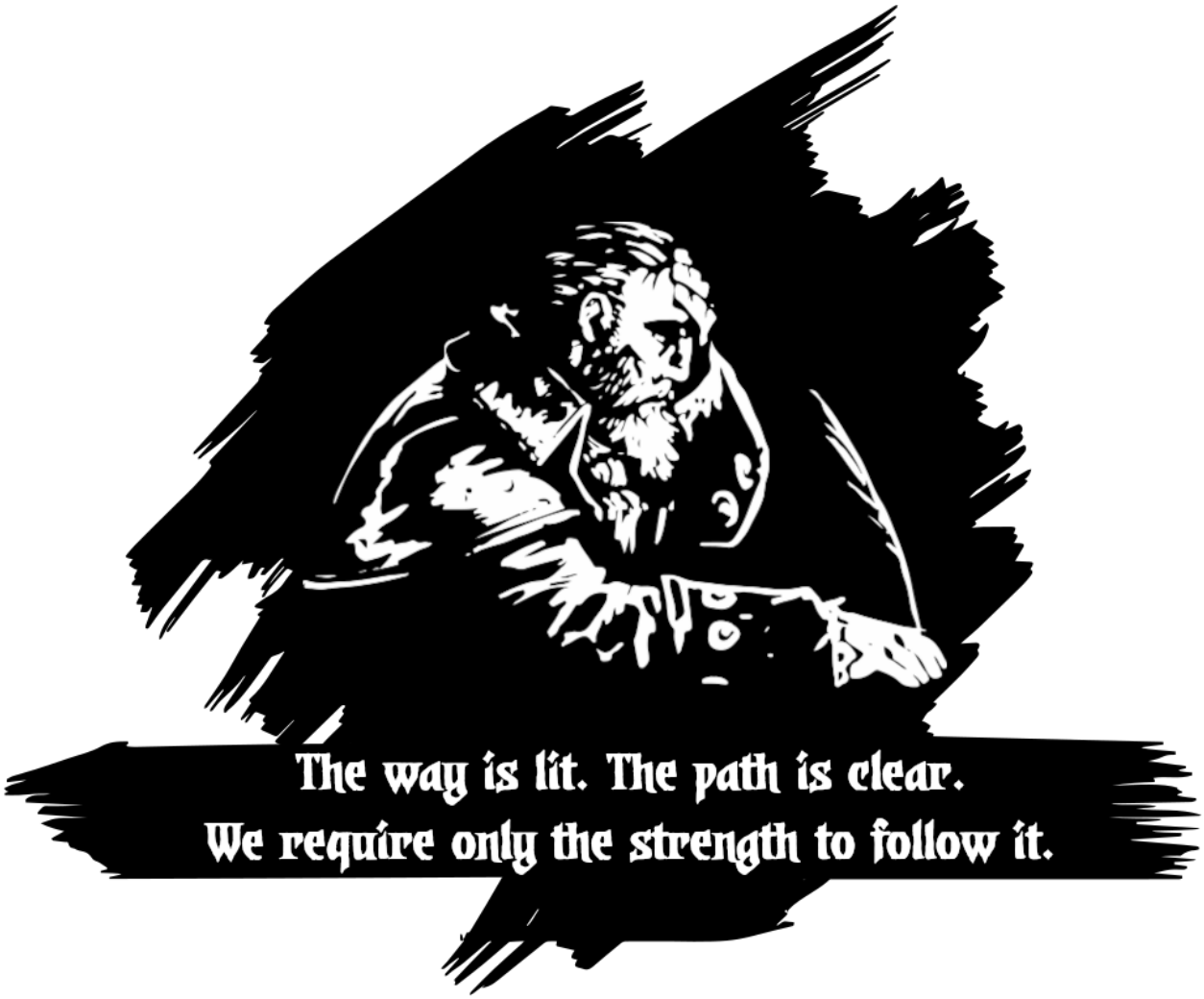
Exercises — Handling Complex

version **#7be580532266ed398481e31366afcc24b1950c2a**



Assistants C/Unix 2023 <assistants@tickets.assistants.epita.fr>

Copyright

This document is for internal use at EPITA (website) only.

Contents

*https://intra.assistants.epita.fr

File Tree

```
handling_complex/
├── complex.h
├── complex_operations.c  (to submit)
├── complex_operations.h
├── complex_print.c  (to submit)
└── complex_print.h
```

Authorized functions : You are only allowed to use the following functions

- printf(3)

Authorized headers : You are only allowed to use the functions defined in the following headers

- err.h
- errno.h
- assert.h
- stddef.h

Compilation : Your code must compile with the following flags

- -std=c99 -pedantic -Werror -Wall -Wextra -Wvla

Main function : None

1 Introduction

In this exercise, we will write a set of functions to manage complex numbers. The goal is to learn to handle structures.

1.1 Structure

```c
struct complex
{
    float real;
    float img;
};
```

The floats `real` and `img` represent respectively the real part and the imaginary part of a complex number.

The declaration of the structure is given in `complex.h`. You **must** not modify this file.

2 Functions to implement

2.1 print_complex

```
void print_complex(struct complex a);
```

The function `print_complex` takes a complex number as argument and writes this number under the form of:

```
complex(<real> + <imaginary>i)
```

If its imaginary part is negative, then the output must be:

```
complex(<real> - <abs(imaginary)>i)
```

For example, if its real part is `2f` and its imaginary part is `-1f`, your complex will be printed like this:

```
complex(2.00 - 1.00i)
```

However, if its real part is `-1f` and its imaginary part is `2f`, your complex will be printed like this:

```
complex(-1.00 + 2.00i)
```

The output **must** be followed by a *newline character*.

Note that two digits must be displayed after the comma. Also, zeroes should be displayed, even if the number is `0.00`.

2.2 neg_complex

```
struct complex neg_complex(struct complex a);
```

The function `neg_complex` takes a complex number as argument and returns the corresponding opposite complex number.

Tips

The opposite of a complex number $z = a + bi$, written $-z$, is computed like this:

$$-z = -a - bi$$

2.3 add_complex

```
struct complex add_complex(struct complex a, struct complex b);
```

The function `add_complex` takes two complex numbers as argument and returns a complex number, which is the addition of the two arguments.

Tips

Let $z_1 = a+bi$ and $z_2 = c+di$ two complex numbers. The sum of two complex numbers is calculated like this:

$$z1 + z2 = (a + c) + i \times (b + d)$$

2.4 sub_complex

```
struct complex sub_complex(struct complex a, struct complex b);
```

The function `sub_complex` takes two complex numbers as argument and returns a complex number, which is the subtraction of the two arguments.

Tips

Let $z_1 = a + bi$ and $z_2 = c + di$ two complex numbers. The subtraction of two complex numbers is calculated like this:

$$z1 - z2 = (a - c) + i \times (b - d)$$

2.5 mul_complex

```
struct complex mul_complex(struct complex a, struct complex b);
```

The function `mul_complex` takes two complex numbers as argument and returns a complex number, which is the multiplication of the two arguments.

Tips

Let $z_1 = a + bi$ and $z_2 = c + di$ two complex numbers. The multiplication of two complex numbers is calculated like this:

$$z_1 * z_2 = (a \times c - b \times d) + i \times (a \times d + c \times b)$$

2.6 div_complex

```
struct complex div_complex(struct complex a, struct complex b);
```

The function `div_complex` takes two complex numbers as argument and returns a complex number, which is the first argument divided by the second.

Note: You do not have to handle divisions by 0.

Tips

Let $z_1 = a + bi$ and $z_2 = c + di$ two complex numbers. The division of two complex numbers is calculated like this:

$$\frac{z_1}{z_2} = \frac{(a \times c + b \times d) + i \times (b \times c - a \times d)}{c^2 + d^2}$$

The way is lit. The path is clear. We require only the strength to follow it.