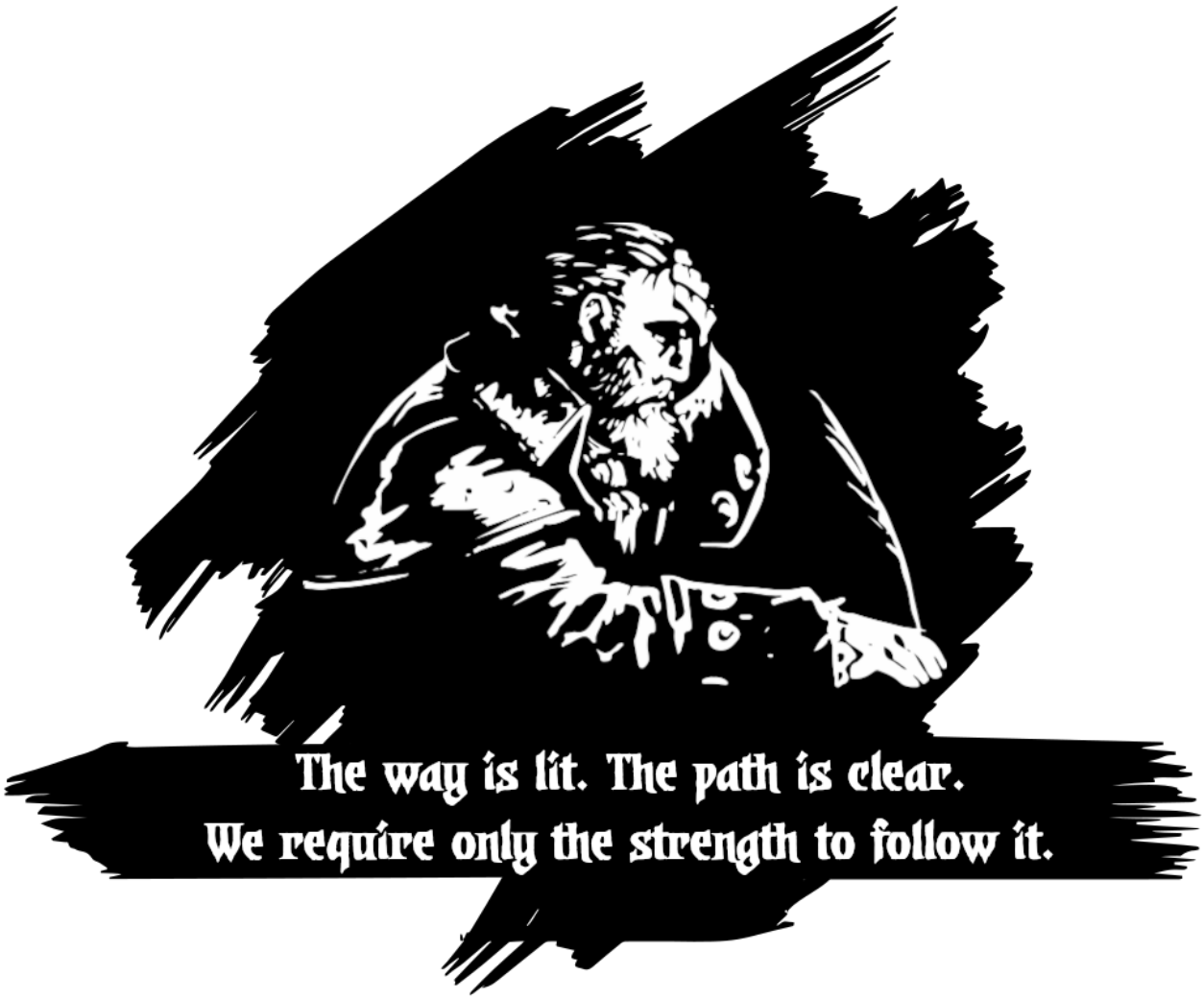
Exercises — Insertion Sort

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

```
insertion_sort/
├── insertion_sort.c   (to submit)
└── insertion_sort.h   (to submit)
```

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

- malloc(3)
- calloc(3)
- free(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 Goal

You have to implement a *generic* insertion sort. It will take two arguments: an array of pointers and a **comparison function**. The array of pointers is **NULL** terminated to indicate the end. The function prototype is:

```
void insertion_sort(void **array, f_cmp comp);
```

The argument `comp` is a pointer to a comparison function which returns an integer lower, equal or greater than zero if the first argument is respectively lower, equal or greater than the second argument. For example, the function `strcmp(3)` matches this description. Here is the definition of f_cmp :

```
typedef int (*f_cmp)(const void *, const void *);
```

Your implementation performance will be tested: don't try to trick us with a simple bubble sort.

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