Exercises — Heap Sort

version **#7be580532266ed398481e31366afcc24b1950c2a**

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

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

Copyright

Contents

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

File Tree

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

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

In this exercise you will have to implement two algorithms about the heap data structure: `heapify` and `heap_sort`.

The main idea behind `heapify` is to construct a max-heap. A max-heap is a complete binary tree whose biggest value is at the root and in which each children is smaller than its parent.

`heap_sort` is an algorithm to sort a list by using `heapify`. You need to swap the root of the heap with the last element of it, placing the biggest element of the array at the end. Then, you need to transform again the `n - 1` first elements of the heap repeat the process on them until you get the list sorted.

2 Heapify

Write the following function:

```c
void heapify(int *array, size_t size);
```

This function transforms in place an integer array to a max-heap. It is useful when you need to update a list after swapping elements. For this exercise, you can assume that `array` will never be `NULL`.

3 Sort

Write the following function:

```
void heap_sort(int *array, size_t size);
```

This function sorts the elements of the input integer array in place by ascending order using the `heap sort` algorithm. For this exercise, you can assume that `array` will never be `NULL`.

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