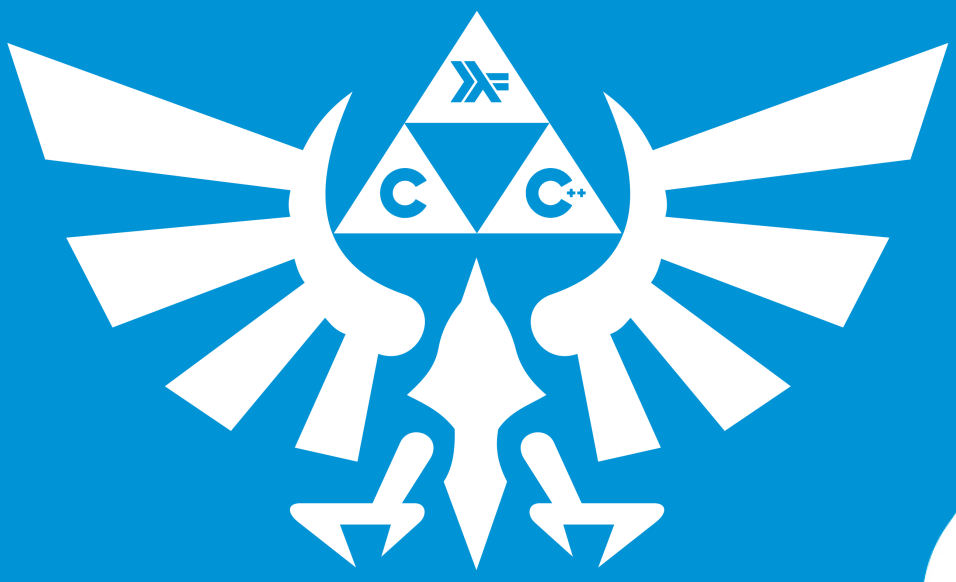
{EPITECH}

# RUSH 1

## PUSHSWAP-CHECKER



# RUSH 1

**binary name:** pushswap_checker
**language:** haskell
**compilation:** via Makefile, including re, clean and fclean rules
**build tool:** free (stack recommanded)

- ✓ The totality of your source files, except all useless files (binary, temp files, objfiles,...), must be included in your delivery.
- ✓ All the bonus files (including a potential specific Makefile) should be in a directory named bonus.
- ✓ Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

A few years ago at Epitech, we had a project called **push_swap**. The goal was to sort a list of integers using two stacks and a limited set of operations. This project was an excellent way to learn how to manipulate data structures and optimize algorithms. It has since been replaced by the project Organized.

The **pushswap_checker** is a project that aims to verify the correctness of the **push_swap** project, similar to how the moulinette works. It is a great opportunity to learn how to create testing tools and to deepen your algorithmic skills.

The goal of this rush is to produce a program capable of checking if a sequence of **push_swap** operations effectively sorts a list of integers.

This is your first project in Haskell, so reflect on what you have learned over the past few days

# Instructions

Your program will take a list of signed integers on the command line, and read a single line of operations separated by spaces from the standard input.

Your program will display “OK” on the standard output (followed by a newline) if the final result is a sorted list (and the second empty), or “KO” otherwise.

If the result is negative, you are free to display additionnal informations after the “KO” string.

In case of inadequate input, your program will return 84. Otherwise, it will return 0.

**Examples:**

```
Terminal
~/B-PDG-300> echo ‘‘sa pb pb pb sa pa pa pa’’ | ./pushswap_checker 2 1 3 6 5 8
OK
~/B-PDG-300> echo ‘‘sa pb pb pb’’ | ./pushswap_checker 2 1 3 6 5 8
KO: ([6,5,8],[3,2,1])
```

# Bonus

As a bonus, you could make a real pushswap in haskell.

# Reminder of the push_swap project

The game is made up of two lists of numbers named l_a and l_b.
In the beginning, l_b will be empty and l_a will contain a certain amount of positive or negative numbers.
The objective of the game is to sort l_a.

In order to accomplish this, you will only have access to the following operation:

- ✓ sa
  swap the first two elements of l_a (nothing will happen if there aren't enough elements).
- ✓ sb
  swap the first two elements of l_b (nothing will happen if there aren't enough elements).
- ✓ sc
  sa and sb at the same time.
- ✓ pa
  take the first element from l_b and move it to the first position on the l_a list (nothing will happen if l_b is empty).
- ✓ pb
  take the first element from l_a and move it to the first position on the l_b list (nothing will happen if l_a is empty).
- ✓ ra
  rotate l_a toward the beginning, the first element will become the last.
- ✓ rb
  rotate l_b toward the beginning, the first element will become the last.
- ✓ rr
  ra and rb at the same time.
- ✓ rra
  rotate l_a toward the end, the last element will become the first.
- ✓ rrb
  rotate l_b toward the end, the last element will become the first.
- ✓ rrr
  rra and rrb at the same time.

v2