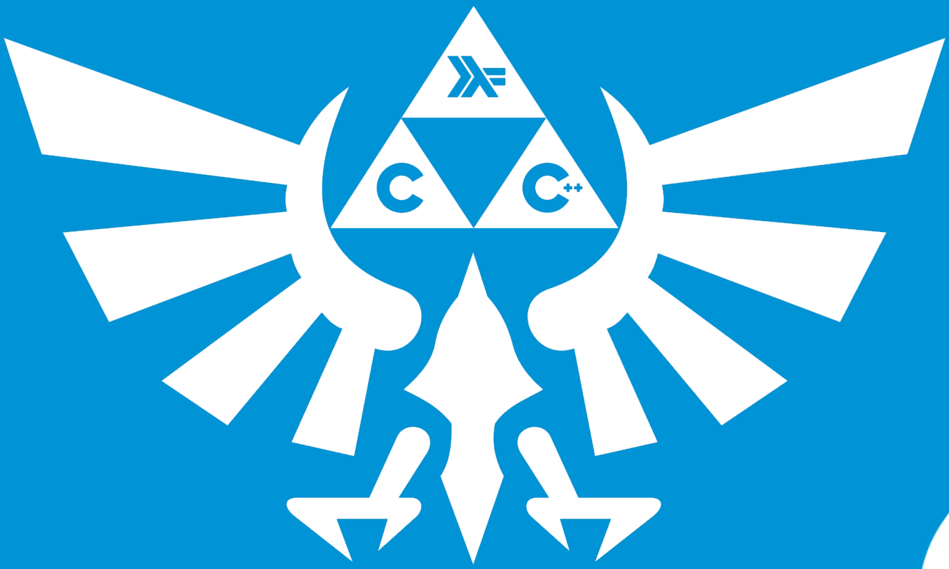
DAY 07 - MORNING

RESISTANCE IS FUTILE



DAY 07 - MORNING

All your exercises will be compiled with `g++` **and the** `-std=c++20 -Wall -Wextra -Werror` **flags**, unless specified otherwise.

All output goes to the standard output, and must be ended by a newline, unless specified otherwise.

None of your files must contain a `main` function, unless specified otherwise. We will use our own `main` functions to compile and test your code. It will include your header files.

For each exercise, the files to turn-in are path relative to the root of the directory. So you **don't** have to put everything in an `exXX` folder.

Read the examples CAREFULLY. They might require things that weren't mentioned in the subject...

The `*alloc`, `free`, `*printf`, `open` and `fopen` functions, as well as the `using namespace` keyword, are forbidden in C++. By the way, `friend` is forbidden too, as well as any library except the standard one.

Unit Tests

It is highly recommended to test your functions as you implement them. It is common practice to create and use what are called **unit tests**.

From now on, we expect you to write unit tests for your functions (when possible). To do so, please follow the instructions in the **"How to write Unit Tests"** document on the intranet, available here.

For them to be executed and evaluated, put a `Makefile` at the root of your directory with the `tests_run` rule as mentionned in the documentation linked above.

Exercise 0 - The Federation

Turn in : Federation.hpp, Federation.cpp, WarpSystem.hpp, WarpSystem.cpp

The **United Planets Federation** is an alliance of people able to travel through space. They all possess the distortion speed – or warp – technology, letting them travel through subspace, and all share common values.

Starfleet is an organization tightly coupled to the **Federation**. Its primary mission is to collect as much information as possible about the **Universe** (and life and everything). The fleet also has a defensive purpose (which is why all their vessels are prepped and armed), which can turn offensive if need be.

You must create a `Federation` namespace, which contains all the elements that allow the **Federation** to exist. Within the `Federation` namespace, create a nested `Starfleet` namespace. It contains a `Ship` class, which will be used to create spaceships.

Each `Ship` must have the following attributes :

```
int _length;
int _width;
std::string _name;
short _maxWarp;
```

These properties must all be provided during the `Ship`'s construction, and cannot be later modified by a method or by directly accessing them.

The class' constructor must have the following prototype :

```
Ship(int length, int width, std::string name, short maxWarp);
```

Upon creation, each `Ship` prints the following to the standard output :

```
The ship USS [NAME] has been finished.
It is [LENGTH] m in length and [WIDTH] m in width.
It can go to Warp [MAXWARP]!
```

You must of course replace `[NAME]`, `[LENGTH]`, `[WIDTH]` and `[MAXWARP]` with the approriate values.

Each `Ship` requires a complex system to navigate through space, which you must have to provide. As this system is not exclusive to the **Federation**'s `Ships`, you must create a new `WarpSystem` namespace.