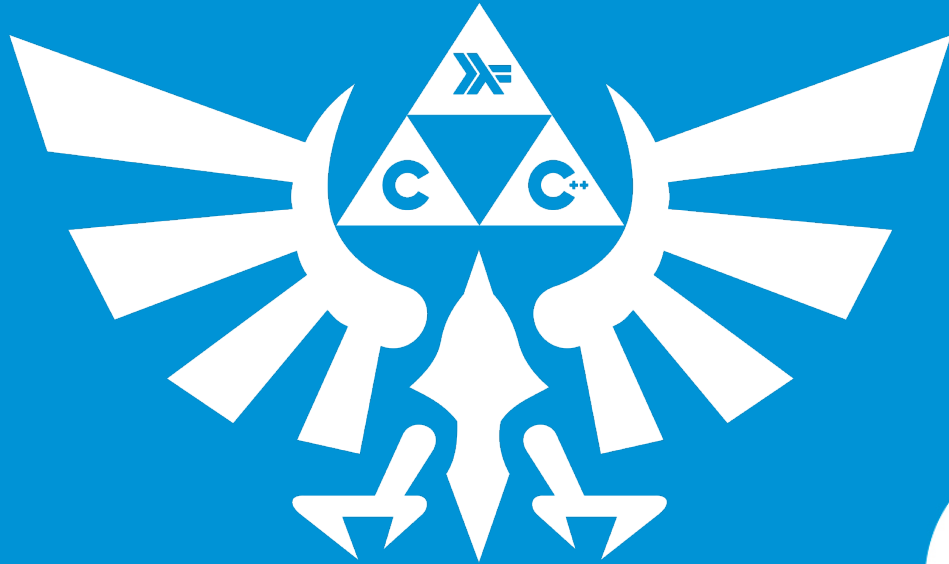




DAY 07 - AFTERNOON

SKAT



DAY 07 - AFTERNOON

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 must be turned-in in a separate directory called `exXX` where XX is the exercise number (for instance `ex01`), unless specified otherwise.



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 mentioned in the documentation linked above.

Exercise 0 - Meeeeeeeddic



Turn in : [Skat.hpp](#), [Skat.cpp](#)

Forbidden features : pointers

Soldiers, welcome to the **SKAT (Special Kreog And Tactical)**. You are here because you want to protect our wonderful country, Australia, from the crawling brood that are **Dingos**. This infamous rot won't rest until they undermine the core foundation of our wonderful country.

I am **drill sergeant Hartog** and I am in charge of making war dogs out of you.

However, you are now at level zero of your life on earth. It's time for you to rack in some experience before you can **show them what you've got**.

Let's start the briefing. As you will very likely get a bullet in your skin on the battlefield, our team of scientists have designed a special package that will save your butt more than you would think: the **stimpak**. This stimpak can repair your bones, suture clean wounds, and so on. As long as you have one left, you have a chance to stay alive and come back home.

Implement the following class :

```
class Skat
{
public:
    Skat(const std::string &name, int stimPaks);
    ~Skat();

    [...] stimPaks();
    const std::string &name();

    void shareStimPaks(int number, [...] stock);
    void addStimPaks(unsigned int number);
    void useStimPaks();
    void status();

private:
    [...]
};
```

A Skat has a name, represented by a [string](#), and a number of stimpaks. By default, your Skat's name is **bob**, and it has 15 stimpaks.

The [stimPaks](#) member function returns the number of stimpaks your Skat currently has. It is possible to modify the number of stimpaks your unit has by calling this function.

The [name](#) member function returns your unit's name. It doesn't modify the calling instance.

The `shareStimpaks` member function lets you provide extra stimpaks to a teammate in need. It increments by `number` the `stock` of stimpaks. After doing so, it decrements your stock by `number` stimpaks. If the number of stimpaks required is too big, print :

```
Don't be greedy
```

On the standard output, and do nothing. Otherwise, print :

```
Keep the change.
```

The `addStimpaks(unsigned int number)` member function adds `number` stimpaks to your unit's collection. If `number` is equal to 0, it prints :

```
Hey boya, did you forget something?
```

Your unit can use stimpaks by calling the `useStimpaks()` member function. It prints :

```
Time to kick some ass and chew bubble gum.
```

When possible. Otherwise, it prints :

```
Mediiiiiiic
```

You can query a unit's status at any point by calling its `status` member function. A unit communicates like so :

```
Soldier [NAME] reporting [NUMBER] stimpaks remaining sir!
```

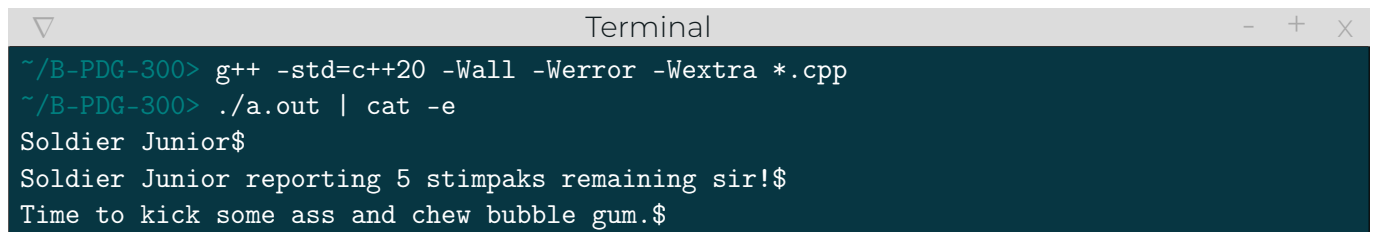
Where `[NAME]` is the name of the unit and `[NUMBER]` its number of stimpaks. This member function can be called on immutable `Skats`.

The following code must compile and display the following output :

```
int main()
{
    Skat s("Junior", 5);

    std::cout << "Soldier " << s.name() << std::endl;
    s.status();
    s.useStimpaks();

    return 0;
}
```



```
~/B-PDG-300> g++ -std=c++20 -Wall -Werror -Wextra *.cpp
~/B-PDG-300> ./a.out | cat -e
Soldier Junior$
Soldier Junior reporting 5 stimpaks remaining sir!$
Time to kick some ass and chew bubble gum.$
```