Contents

1 Getting started **1**

1.1 Answering questions 1

1.2 Warnings 1

2 Part 1 : Homemade sets **2**

3 Part 2 : Object implementation of homemade sets **2**

4 Part 3 : Templates for homemade sets **2**

1 Getting started

1.1 Answering questions

On the terminal, typing only

```
make
```

helps you. The exam is organized into parts, with an increasing difficulty.

You are given `.cpp` and `.hpp` files, where you have to write the code where the comments suggest to do so.

Compiling what you have done is provided my `make`, that calls the compiling command for you. Moreover, it commits the changes into a local git repository, in case of accidental delete... but be sure not to delete files since you are evaluated from the filling of the files we provide.

When you have answered, for example, question 3 in part 1, you can test as many time as you need by the command

```
make part1-question3
```

Do not type everything, rather use the completion key (the `TAB` key).

1.2 Warnings

The documentation is available on this machine, you have no access to internet, and no extra electronic devices are allowed.

DO NOT access collections elements with the `[]` operator, like in `tab[4]`, since this is not efficient within loops.

Each function you will have to implement **is short** (less than 10 lines). Do not get lost in obfuscated code !

2 Part 1 : Homemade sets

Read the `part1.hpp` and `part1.cpp` files, and then read one by one the `part1-question1.cpp`, `part1-question2.cpp`, ... files. Each time, for each question, fill the blanks in `part1.hpp` and `part1.cpp` files. You may need to uncomment lines in the `part1-questionX.cpp` file... follow the instructions given by the comments. Test each question with the

```
make part1-question1
make part1-question2
make part1-question3
...
```

commands.

3 Part 2 : Object implementation of homemade sets

The idea of this section is to gather what has been done before into a class. **Do not modify part1** files, edit the `part2*` files. Of course, you can copy-paste from what you have done previously.

So write `part2.hpp` and `part2.cpp` files as instructed. You may prefer having all the code in `part2.hpp`. In this case, leave `part2.cpp` as it is. Make the `part2-question*` test succeed.

4 Part 3 : Templates for homemade sets

Let us extend what we have done in Part 2 to any type that supports the `<<` operator as well as the `==` operator. **Do not modify part2** files, edit the `part3*` files. No need for `part3.cpp` here.

The file `part3.hpp`, needed to compile the `part3-question*.cpp` tests, is empty. Copy-paste the class definition you have done in `part2.hpp`, and make it be a template.