Remember that the quality of the defenses, hence the quality of the of the school on the labor market depends on you. The remote defences during the Covid crisis allows more flexibility so you can progress into your curriculum, but also brings more risks of cheat, injustice, laziness, that will harm everyone's skills development. We do count on your maturity and wisdom during these remote defenses for the benefits of the entire community.

SCALE FOR PROJECT FT_CONTAINERS (/PROJECTS/FT_CONTAINERS)

You should evaluate 1 student in this team

★

Git repository

git@vogsphere-v2.42sp.org.br:vogsphere/intra-uuid-f3a03c53-fb26-4bc0 📋

Introduction

- Only grade the work that is in the student or group's
GiT repository.

- Double-check that the GiT repository belongs to the student
or the group. Ensure that the work is for the relevant project
and also check that "git clone" is used in an empty folder.

- Check carefully that no malicious aliases were used to fool you
and make you evaluate something other than the content of the
official repository.

- To avoid any surprises, carefully check that both the evaluating
and the evaluated students have reviewed the possible scripts used
to facilitate the grading.

- If the evaluating student has not completed that particular
project yet, it is mandatory for this student to read the
entire subject prior to starting the defense.

- Use the flags available on this scale to signal an empty repository,
non-functioning program, a norm error, cheating etc. In these cases,
the grading is over and the final grade is 0 (or -42 in case of
cheating). However, except for cheating, you are
encouraged to continue to discuss your work (even if you have not
finished it) to identify any issues that may have caused
this failure and avoid repeating the same mistake in the future.

- Remember that for the duration of the defense, no segfault,

no other unexpected, premature, uncontrolled or unexpected
termination of the program, else the final grade is 0. Use the
appropriate flag.
You should never have to edit any file except the configuration file if it exists.
If you want to edit a file, take the time to explicit the reasons with the
evaluated student and make sure both of you are okay with this.

- You must also verify the absence of memory leaks. Any memory allocated on the heap must
be properly freed before the end of execution.
You are allowed to use any of the different tools available on the computer, such as
leaks, valgrind, or e_fence. In case of memory leaks, tick the appropriate flag.

Disclaimer

Please respect the following rules:

- Remain polite, courteous, respectful and constructive
throughout the evaluation process. The well-being of the community
depends on it.

- Identify with the person (or the group) evaluated the eventual
dysfunctions of the work. Take the time to discuss
and debate the problems you have identified.

- You must consider that there might be some difference in how your
peers might have understood the project's instructions and the
scope of its functionalities. Always keep an open mind and grade
him/her as honestly as possible. The pedagogy is valid only and
only if peer-evaluation is conducted seriously.

Guidelines

You must compile with clang++, with -Wall -Wextra -Werror
As a reminder, this project is in C++98.
C++11 (and later) members functions or containers are NOT expected.

Any of these means you must not grade the exercise in question:
- A function is implemented in a header (except in a template)
- A Makefile compiles without flags and/or with something other than clang++

Any of these means that you must flag the project as Forbidden Function:
- Use of a "C" function (*alloc, *printf, free)
- Use of a function not allowed in the subject
- Use of "using namespace" or "friend" (in this subject, "friend" is allowed on specific occasions).
- Use of an external library, or C++20 features
- Use of an already existing container, or any existing function, to implement another container

---

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/32341/en.subject.pdf)

main.cpp (/uploads/document/document/5310/main.cpp)

Containers check

Verify that each container is correctly implemented.

Vector basics

Make sure that each member function, overload and iterator are present and work as expected.
Use the standard library container to check that everything works the same way.

☑ Yes   ✕ No

Vector Advance

The inner data structure should be a dynamic array.
const_iterator and iterators should be comparable.
Check the dynamic reallocation system.
Test the swap function.
All iterators, pointers and references referring to elements in both containers remain valid,
and are now referring to the same elements they referred to before the call,
but in the other container, where they now iterate.
Check that friend keyword is used only for operators.

☑ Yes   ✕ No

Vector Performance

Make sure that the speed is reasonable compared to the STL vector!
For example, a deep copy should allocate all the memory in one call.

☑ Yes   ✕ No

Map Basics

Make sure that each member function, overload and non-member overload are present and work as expected.
Use the standard library container to check that everything works the same way.

☑ Yes   ✕ No

Map Advance

Check the inner structure should be an ordered tree (AVL tree, R-B tree...)
Check that pair<> is recoded and used.
ft::make_pair worked as intended.
There are never two of the same Key in one map.
Check that the container is ordered.
Check std::allocator and allocator::rebind are used and there's no direct usage of new.
Check that insert or delete do not invalidate iterators.

Swap function should not move data but only pointers.
Friend keyword should only be used for operator overload.
There's no memory leak.

☑ Yes   ☒ No

Map Performance

Make sure that the speed is reasonable compared to the STL!
Slower than the STL is ok.
A complete timeout is not.
If it's more than x20 slower than the STL you should count false.

☑ Yes   ☒ No

Stack Basics

Make sure that each member function, overload and non-member overload are present and work as expected.
Use the standard library container to check that everything works the same way.

☑ Yes   ☒ No

Stack Advance

The standard container classes vector, deque and list are compatible as underlying container.
The stack cannot be iterate.
The underlying container must be protected and not private!

☑ Yes   ☒ No

Bonus

If and only if the mandatory part is complete, you can go on and grade the bonus containers asked in the subject.

Set

Make sure that each member function, overload and non-member overload are present and work as expected.
Use the standard library container to check that everything works the same way.
The inner data structure must be a Black and Red tree.
Ask the student details.
How it works ? if you have any doubt don't count this bonus a std::set is easier than a map.
The only bonus here is for Red and Black tree!

☑ Yes   ☒ No

Ratings

Don't forget to check the flag corresponding to the defense

✔ Ok   ★ Outstanding project

Conclusion

Leave a comment on this evaluation

Finish evaluation

Privacy policy
(https://signin.intra.42.fr/legal/terms/5)

Terms of use for video surveillance
(https://signin.intra.42.fr/legal/terms/1)

Rules of procedure
(https://signin.intra.42.fr/legal/terms/4)

Declaration on the use of cookies
(https://signin.intra.42.fr/legal/terms/2)

General term of use of the site
(https://signin.intra.42.fr/legal/terms/6)

Legal notices
(https://signin.intra.42.fr/legal/terms/3)