













Menu

Your projects

Introduction

Git repository

Please comply with the following rules:

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

SCALE FOR PROJECT FT_CONTAINERS You should evaluate 1 student in this team

git@vogsphere-v2-bg.1

- Identify with the student or group whose work is evaluated the possible dysfunctions in their project. Take the time to discuss and debate the problems that may have been identified.
- You must consider that there might be some differences in how your peers might have understood the project's instructions and the scope of its functionalities. Always keep an open mind and grade them as honestly as possible. The pedagogy is useful only and only if the peer-evaluation is done seriously.

Guidelines

- Only grade the work that was turned in the Git repository of the evaluated student or group.
- Double-check that the Git repository belongs to the student(s). Ensure that the project is the one expected. Also, check that 'git clone' is used in an empty folder.
- Check carefully that no malicious aliases was used to fool you and make you evaluate something that is not the content of the official repository.
- To avoid any surprises and if applicable, review together any scripts used to facilitate the grading (scripts for testing or automation).
- If you have not completed the assignment you are going to evaluate, you have to read the entire subject prior to starting the evaluation process.
- Use the available flags to report an empty repository, a non-functioning program, a Norm error, cheating, and so forth. In these cases, the evaluation process ends and the final grade is 0, or -42 in case of cheating. However, except for cheating, student are strongly encouraged to review together the work that was turned in, in order to identify any mistakes that shouldn't be repeated in the future.
- 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.

Attachments

subject.pdf main.cpp

Preliminary tests

If cheating is suspected, the evaluation stops here. Use the "Cheat" flag to report it. Take this decision calmly, wisely, and please, use this button with caution.

Prerequisites

The code must compile with c++ and the flags -Wall -Weytra -Werrer

me code mosi compile willi c · · dila me mags - + + an - + + exila - + + emon.

There is a Makefile, the project compiles correctly with the required options, is written in C++.

The code should also compile with the flag -std=c++98

Don't forget this project has to follow the C++98 standard. Thus, C++11 (and later) 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 file (except for template functions).
- A Makefile compiles without the required flags and/or another compiler than c++.

Any of these means that you must flag the project with "Forbidden Function":

- Use of a "C" function (*alloc, *printf, free).
- Use of a function not allowed in the exercise guidelines.
- Use of "using namespace" or the "friend" keyword (in this project, the "friend" keyword is allowed for specific uses, you'll see).
- Use of an external library, or features from versions other than C++98.
- Use of a STL container to implement another container.



Mandatory part

Verify that each mandatory container is correctly implemented. If the vector or the map is missing, check the 'Incomplete work' flag. But note the project can be passed without the stack. Do not hesitate to ask to the evaluated student what is the reference they used. A main.cpp is available to download on this page. It should compile with the student's containers. The evaluated student has also to turn in their own tests.

Vector - Basics

Make sure that every member function, non-member function and overload is present and works as expected. Same thing for iterators.

Use the STL container to check that everything works the same way.

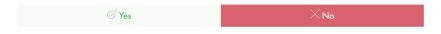


Vector - Advance

- 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:

After swap, all the iterators, pointers and references pointing to elements in both containers remain valid and are now pointing to the same elements than before the call but in the other container (where they now iterate).

- Check that the friend keyword is used only for the relational operators.



Vector - Performance

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



Map - Basics

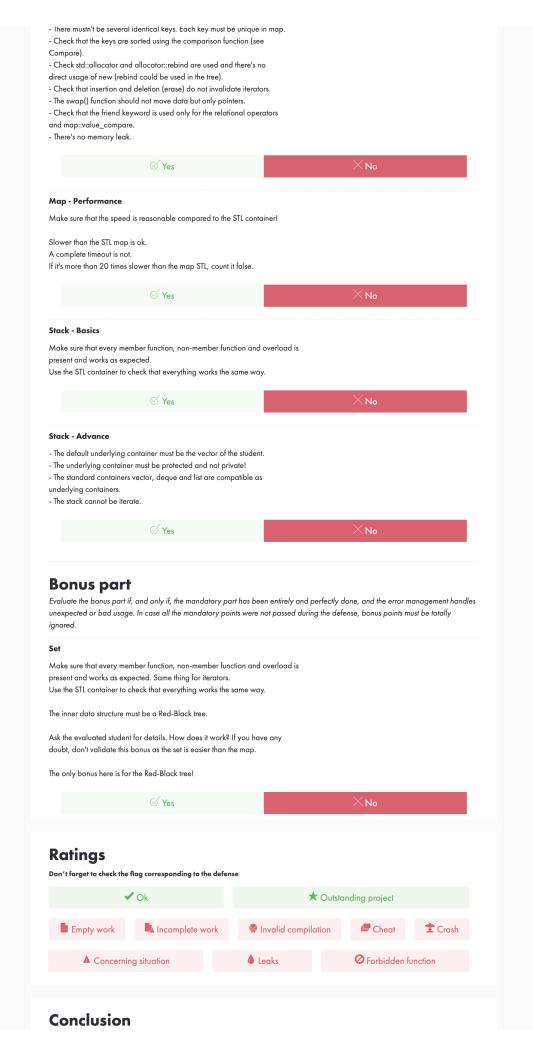
Make sure that every member function, non-member function and overload is present and works as expected. Same thing for iterators.

Use the STL container to check that everything works the same way.



Map - Advance

- Check the inner structure. It should be an ordered tree (AVL tree, Red-Black tree, and so forth). For example, a simple array is not ok.
- Verify that pair<> is recoded and used.
- Verify that ft::make_pair() behaves as expected.



	Leave a comment on th	is evaluation				
			Finish evaluation			
T	Terms of use for video surveillance	Rules of procedure	Declaration on the use of cookies	General term of use of the site	Legal notices	Privacy polic