SCALE FOR PROJECT LIBFTPP

You should evaluate 1 student in this team

Git repository

`git@vogsphere.42perpignan.fr:vogsphere/intra-uuid-417ad33`

Introduction

- 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

- 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 was 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 defence.

- 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, with the exception of cheating, you are encouraged to continue to discuss your work (even if you have not finished it) in order 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 defence, 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.

Attachments

subject.pdf libftpp.1.0.tar.xz

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 -Wextra -Werror`.
Don't forget this project has to follow at least the C++11 standard. Thus, C++13 (and later) functions or containers are accepted.

Any of these issues 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 with a compiler other than `c++`.

Any of these issues 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 an external library, such as Boost, is forbidden.

[✓ Yes] No

Mandatory Part

Pool class and Pool::Object nested class

The evaluated student has been provided with `main_pool.cpp`, which you can also download from the correction page.
The evaluated student may have edited the way the `Pool` or `Pool::Object` works, so you are allowed to edit the `main` function as long as it still respects the structure of the provided `main`.

Once edited, the `main` should output what is indicated inside the function.

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

DataBuffer class

The evaluated student has been provided with `main_data_buffer.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way the `DataBuffer` works, so you are not allowed to edit the `main` function.

Once executed, the `main` should output what is indicated inside the function.

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

Memento class

The evaluated student has been provided with `main_memento.cpp`, which you can also download from the correction page.

In this main, you should find a `TestClass`, inheriting from `Memento`.
This class must implement the two methods created by the evaluated student to push and retrieve information from a snapshot.

As the student may have changed the way information is inserted into it, you are allowed to edit the `main` function to match its implementation.

Once edited, you should be able to compile and execute the code, resulting in the output described inside the function.

If the output is indeed as expected, you should consider validating this question.

On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

Observer class

The evaluated student has been provided with `main_observer.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way the `Observer` works, so you are not allowed to edit the `main` function.

Once executed, the `main` should output what is indicated inside the function.

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

Singleton class

The evaluated student has been provided with `main_singleton.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way the `Observer` works, so you are not allowed to edit the structure of the `main` function.

However, they may have changed the exceptions, so it's acceptable if the messages or exception types differ from the original (You may change the `const std::exception&` inside the `catch` block to any other type of exception).

Once executed, the `main` should output what is indicated inside the function.

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

StateMachine class

The evaluated student has been provided with `main_state_machine.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way the `StateMachine` works, so you are not allowed to edit the structure of the `main` function. However, they may have changed the exceptions, so it's acceptable if the messages or exception types differ from the original (You may change the `const std::invalid_argument&` inside the `catch` block to any other type of exception).

Once executed, the `main` should output what is indicated inside the function.

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

ThreadSafeIOStream class

The evaluated student has been provided with `main_thread_safe_iostream.cpp`, which you can also download from the correction page.
The evaluated student may have customized the prefix of the `ThreadSafeIOStream` class, such as adding additional information, so the resulting output may differ from what is expected. However, as long as it still outputs what the code expects, you may consider this as valid.
Obviously, the two threads created inside the `main` must have output their information with at least their prefix to differentiate which line is output by whom.

If the output is coherent, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

ThreadSafeQueue class

The evaluated student has been provided with `main_thread_safe_queue.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way the `ThreadSafeQueue` works, so you're not allowed to edit the structure of the `main`, but they may have changed the exceptions. Thus, it is allowed to have different messages or exception types (you may change `const std::runtime_error&` inside the catch block to any other type of exception).

Once executed, the `main` should output what is indicated within the `main`.

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

As required, the evaluated student must have found a way to allow access to a `threadSafeCout` variable from anywhere in the code, provided you include `thread_safe_iostream.hpp`. The name of the variable may differ, so be prepared to possibly edit some `main` code later to match the name chosen by the evaluated student.
If they have done this, you should consider giving this exercise a score of 0 and stop the evaluation here.

Yes [✓ No]

Thread class

The evaluated student has been provided with `main_thread.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way the `Thread` works, so you're not allowed to edit the structure of the `main`.

Once executed, the `main` should output what is indicated within the `main`.
Output messages must be complete and not disturbed by other threads while being outputted to the console.

Upon closing the program, the threads must be terminated cleanly by the program.
If they are closed forcefully by the program, you should consider giving this question a score of 0 and continue the evaluation.

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

WorkerPool class

The evaluated student has been provided with `main_worker_pool.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way the `WorkerPool` works, so you're not allowed to edit the structure of the `main`.

Once executed, the `main` should output what is indicated within the `main`.
Output messages must be complete and not disturbed by other threads while being outputted to the console.

Upon closing the program, the threads must be terminated cleanly by the program.
If they are closed forcefully by the program, you should consider giving this question a score of 0 and continue the evaluation.

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

PersistentWorker class

The evaluated student has been provided with `main_persistent_worker.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way the `PersistentWorker` works, so you're not allowed to edit the structure of the `main`.

Once executed, the `main` should output what is indicated within the `main`.
Output messages must be complete and not disturbed by other threads while being outputted to the console.

Upon closing the program, the threads must be terminated cleanly by the program.
If they are closed forcefully by the program, you should consider giving this question a score of 0 and continue the evaluation.

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

Network classes

The evaluated student has been provided with two separate `main` files:

- `main_server.cpp`, which is supposed to create, start, and manage a server hosted on port 8080.
- `main_client.cpp`, which is supposed to create, connect, and manage a client connected to the previous server.

The evaluated student may need to modify the `main` functions in order to make their classes work properly.
Once edited, you should be able to compile and execute the code, resulting in the output described within the `main` files.

You must compile both executables and run them, starting with the server, in two different console windows.

Upon execution, you must obtain the following output:

- **Client side:** 1) "Message sent. Press enter to exit." 2) "Received a doubled value: 84."

Once the doubled value message is received, you must press enter to close the client side cleanly by disconnecting it first.

- **Server side:** 1) "Receiving an int 42 from client [Client ID may differ]." 2) "Receiving a string 'Hello' of length 5 from client [Client ID may differ]."

If the output is indeed as expected, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

IVector2 class

The evaluated student has been provided with `main_vector_2.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way `IVector2` works, so you are not allowed to modify the structure of the `main`.

Once executed, the `main` should produce the output as indicated within the `main`.

If the output matches the expected results, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

IVector3 class

The evaluated student has been provided with `main_vector_3.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way `IVector3` works, so you are not allowed to modify the structure of the `main`.

Once executed, the `main` should produce the output as indicated within the `main`.

If the output matches the expected results, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

Random2DCoordinateGenerator class

The evaluated student has been provided with `main_random_2D_coordinate_generator.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way `Random2DCoordinateGenerator` works, so you are not allowed to modify the structure of the `main`.

Once executed, the `main` should produce the output as indicated within the `main`.

If the output matches the expected results, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

PerlinNoise2D class

The evaluated student has been provided with `main_perlin_2D.cpp`, which you can also download from the correction page.
The evaluated student should not have edited the way `PerlinNoise2D` works, but you are allowed to modify the calculations used to obtain samples from the Perlin noise generator.

Once executed, the `main` should produce the output as indicated within the `main`.

Perlin values should fluctuate "smoothly" depending on the factor used for sampling, resulting in output without abrupt changes, resembling a map view from above.

If the output matches the expected results and shows smooth fluctuations, you should consider validating this question.
On the other hand, if the evaluated student fails to produce the correct output or the output does not show smooth fluctuations, you should consider giving this question a score of 0 and continue the evaluation.

Yes [✓ No]

Bonus

We will consider bonus features only if the mandatory part is rated as excellent. This means that the evaluated student must have completed the mandatory part from start to finish. If the mandatory part did not achieve a perfect score during this evaluation, any bonus features will be completely disregarded.

Additionnal features

Ask the evaluated student if their library includes any additional classes beyond the mandatory requirements.
For each relevant additional class provided, award one point in this section.

Rate it from 0 (failed) through 5 (excellent)

○

Ratings

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

✓ Ok | ★ Outstanding project

Empty work | Incomplete work | Invalid compilation | Norme | Cheat | Crash

Incomplete group | Concerning situation | Leaks | Forbidden function

Can't support / explain code

Conclusion

Leave a comment on this evaluation

☐

Finish evaluation