SCALE FOR PROJECT CPP MODULE 06 (/PROJECTS/CPP-MODULE-06)

You should evaluate 1 student in this team

★

Git repository

`git@vogsphere-v2.42madrid.com:vogsphere/intra-uuid-9d22b2`

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 before 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, 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 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"
- Use of an external library, or C++11 (and later) features

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/33554/en.subject.pdf)

Exercise 00: Scalar conversion

This exercise shows usage of the static_cast.

Scalar conversion

Is the program running as intended, and did the student use the static_cast to convert values? We'll accept the use of implicit casts for promotion casts only.

Anyway, please don't be too uncompromising
towards the exercise's outputs if the spirit of the exercise is respected.
Failing this exercise does not prevent grading the next ones.

☑ Yes ☒ No

Exercise 01: Serialization

This exercise shows usage of the reinterpret_cast.

Retyping of raw data

Is the program running as intended?
reinterpret_cast<> should be used twice
Once from data* to uintptr_t.
The second from uintptr_t to data*.
And the final data struct should be usable.

☑ Yes ☒ No

Exercice 02: Identify real type

This exercise shows usage of the dynamic_cast.

Real type identification

Is the program running as intended?
Check the code and did the student use the
dynamic_casts to identify the real type?
void identify(Base* p) should check if the cast return is NULL.
void identify(Base\& p) should use a try-catch to check if the cast failed.
(In case you're wondering, the header must not appear anywhere.)

☑ Yes ☒ No

Ratings

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

✔ Ok ★ Outstanding project

Empty work No author file Invalid compilation Norme Cheat Crash

Leaks

Forbidden function

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)