☰ Q (https://profile.intra.42.fr/searches)

SCALE FOR PROJECT CPP MODULE 00 (/PROJECTS/ CPP-MODULE-00)

You should evaluate 1 student in this team

★

Git repository

git@vogsphere.42paris.fr:vogsphere/intra-uuid-5a58943c-d5c8-45e9-b86ː 📋

Introduction

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.

- 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 (https://cdn.intra.42.fr/pdf/pdf/168834/en.subject.pdf)

☐ Account.hpp (https://cdn.intra.42.fr/document/document/35876/Account.hpp)

☐ 19920104_091532.log (https://cdn.intra.42.fr/document/document/35877/19920104_091532.log)

☐ tests.cpp (https://cdn.intra.42.fr/document/document/35878/tests.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 -Wextra -Werror
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 <ns_name>" or the "friend" keyword.
- Use of an external library, or features from versions other than C++98.

⌀ Yes   ✕ No

Exercise 00: Megaphone

This exercise is a warm-up to discover basic C++ I/O streams.

Makefile

There is a Makefile that compiles using the appropriate flags.
If this is not the case, the evaluation of this exercise stops here. you can move on to the next exercise.

⊘ Yes   ✕ No

Is it working?

This exercise is about developing a to_upper program with a specific
behavior when run without any parameter. This has to be solved in a C++
approach (strings/upper).

⊘ Yes   ✕ No

Exercise 01: My Awesome Phonebook

*This exercise is about writing simple classes and a small interactive program that uses them. If the exercise is not
fully functional, grade what can be graded.*

Makefile

There is a Makefile that compiles using the appropriate flags.
If this is not the case, the evaluation of this exercise stops here. you can move on to the next exercise.

⊘ Yes   ✕ No

Error handling

This exercise requires a few error handling but there are no expected behaviors in the subject. Quitting or other
handling is fine. Segfault is not! :D

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

5

The EXIT command

Rate the EXIT command as described in the subject.

⊘ Yes   ✕ No

Visibility

The attributes of the class Contact should be private. The class should
expose the corresponding accessors. Also, check that anything that will
always be used inside a class (not only in the Contact class) is private,
and that anything that can be used outside a class is public. Beginners
tend to put everything in public, that's not what you want here!

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

5

The Contact and the Phonebook classes

The code must have a Contact class (or whatever name the student gave).
This class must have attributes for each contact fields. There also must

be a Phonebook class containing an array of Contacts.

$\oslash$ Yes $\times$ No

Read/Eval loop

The program must have a kind of read/eval loop: reading the input,
processing it, then wait again for another command until an EXIT
command is entered. This loop should be done in a C++ manner (std::cin)!

$\oslash$ Yes $\times$ No

The ADD command

Rate the ADD command as described in the subject.

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

5

The SEARCH command

Rate the SEARCH command as described in the subject. The formatting of the output can be different, it doesn't
matter. This part is about using C++ iomanips and that's what you should focus on.

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

5

Exercise 02: The Job Of Your Dreams

This exercise intends to extract information and directions from useless noise, and to insert new code into an existing context.

Makefile

There is a Makefile that compiles using the appropriate flags.
If this is not the case, the evaluation of this exercise stops here. you can move on to the next exercise.

✅ Yes   ✕ No

Did you save the day?

This exercise is pretty straightforward. Either Account.cpp works, either it does not. Compare the program's output and the provided logs. Any difference (except for the timestamps or the order of the destructors) means the exercise is incorrect.

✅ Yes   ✕ No

Ratings

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

| ✔ Ok | ★ Outstanding project |

Empty work   🔩 Incomplete work   ☠ Invalid compilation   📋 Cheat   ♟ Crash   ⚠ Concerning situation

🌢 Leaks   ⦸ Forbidden function   💬 Can't support / explain code

Conclusion

Leave a comment on this evaluation (2048 chars max)

Finish evaluation