Back to all evaluation sheets

CPP Module 00

You should evaluate 1 student in this team

Introduction

Please follow the rules below:

- ✓ - 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 being 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 if the peer-evaluation is done seriously.

Guidelines

Please follow the guidelines below:

- ✓ - Only grade the work that was turned in to 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 were used to fool you into evaluating 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 must 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 the case of cheating. However, except for cheating, students 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.
- ✓ - Remember that for the duration of the defense, no segfaults or other unexpected, premature, or uncontrolled terminations of the program will be tolerated, 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 explain 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

Please download the attachments below:

- subject.pdf
- Account.hpp
- 19920104_091532.log
- tests.cpp

Mandatory Part

Prerequisites
Suspected Cheating

If cheating is suspected, the grading and evaluation process must end immediately. To report it, select the "Cheat" flag. Be sure to use this option with calm, caution, and discernment.

- ✓ The code must compile with c++ using the flags -Wall -Wextra -Werror.
- ✓ As a reminder, this project must adhere to the C++98 standard. Therefore, C++11 (or other standards) functions and containers are NOT expected.
- ✓ Do not grade the exercise if you find the following:
- ✓ A function implemented in a header file (except for template functions).
- ✓ A Makefile that compiles without the required flags and/or uses something other than c++.
- ✓ Select the "Forbidden function" flag if you encounter:
- ✓ The use of a "C" function (*alloc, *printf, free).
- ✓ The use of a function prohibited in the project.
- ✓ The use of using namespace <ns_name> or the friend keyword.
- ✓ The use of an external library, or features specific to versions later than C++98.

Yes No

Exercise 00 : Megaphone

This exercise is a warm-up to get familiar with I/O in C++.

- ✓ Functionality
- ✓ The goal of this exercise is to develop a `to_upper` function with a specific behavior when launched without parameters.
- ✓ It must be solved using a C++ approach (e.g., using `string` and `toupper`).

Yes No

Exercise 01: My Awesome Phonebook

This exercise is an introduction to writing a simple class and using it in an interactive program. If the exercise doesn't work perfectly, grade what can be evaluated.

✓ Error Handling

✓ There is some error handling required in this program, but specific behaviors are not outlined in the instructions. Exiting the program cleanly or handling errors is acceptable. However, a segmentation fault is not!

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

○ 0 1 2 3 4 5

The EXIT Command

The EXIT Command

✓ Grade the implementation of the `EXIT` command based on the requirements outlined in the exercise instructions.

Yes No

Visibility

Visibility

The attributes of the `Contact` class must be private. The class should

- ✓ provide the appropriate accessors (getters and setters). Additionally, verify that anything used only within a class (not just the `Contact` class) is private, while the rest is public. Beginners often tend to make everything public, so that's what you should check here.

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

0 1 2 3 4 5

The `Contact` Class and the `Phonebook` Class

The `Contact` Class and the `Phonebook` Class

- ✓ - The code must include a `Contact` class (or similarly named class).
- ✓ - The `Contact` class should contain the required attributes.
- ✓ - The code must also include a `Phonebook` class, which contains an array of `Contact` objects.

Yes No

Read/Eval Loop

Read/Eval Loop

- ✓ The program should implement a read/eval loop where it reads and processes input, then waits for new input until it receives the `EXIT` command.
- ✓ The loop should be implemented using C++ conventions (e.g., using `std::cin`).

Yes No

The ADD Command

The ADD Command

✓ Grade the implementation of the `ADD` command based on the requirements outlined in the exercise instructions.

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

[✓ 0] 1 2 3 4 5

The SEARCH Command

The SEARCH Command

✓ Grade the implementation of the `SEARCH` command based on the requirements outlined in the exercise instructions. A slight divergence from the expected format is not critical. The focus here is on the use of C++ "iomanips," so that should be the primary area of evaluation.

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

[✓ 0] 1 2 3 4 5

Exercise 02: The Job of Your Dreams
Did You Save the World?

The goal of this exercise is to retrieve hidden information amid noise and to insert code within an existing context.

✓ This exercise is fairly straightforward. Either `Account.cpp` works, or it doesn't. Compare the program's output with the provided log file. Any difference (except for timestamps or the order of destructors) indicates that the exercise is incorrect.

Yes No

Bonus Part

no bonus

no bonus

✓ no bonus

Yes No

Ratings

OK Outstanding Empty Work

Incomplete Work Invalid Compilation

Cheat Crash Concerning Situations

Leaks Forbidden Functions

Cannot Support/Explain code