42 Evals

Back to all evaluation sheets

CPP Module 09

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 community's well-being depends on it.

Work with the student or group being evaluated to identify potential issues in their project. Take time to discuss and debate the problems identified.

Understand that there may be differences in how peers interpret the project instructions and scope. Always keep an open mind and grade as honestly as possible. Pedagogy is effective only when peer evaluations are taken seriously.

Guidelines

Please follow the guidelines below:

Only grade the work submitted to the Git repository of the evaluated student or group.

Double-check that the Git repository belongs to the student(s) and that the project is the one expected. Ensure that git clone is used in an empty folder.

Carefully verify that no malicious aliases are used to deceive the evaluator into grading non-official content.

If applicable, review any scripts used for testing or automation together with the student.

If you haven't completed the assignment you're evaluating, read the entire subject before starting the evaluation.

Use the available flags to report an empty repository, a non-functioning program, a Norm error, or cheating. The evaluation process ends with a final grade of 0 (or -42 for cheating). However, except in cases of cheating, students are encouraged to review the work together to identify mistakes to avoid in the future.

Remember that no segfaults or other unexpected program terminations will be tolerated during the evaluation. If this occurs, the final grade is 0. Use the appropriate flag.

You should not need to edit any files except the configuration file, if it exists. If editing a file is necessary, explain the reasons to the evaluated student and ensure mutual agreement.

Verify the absence of memory leaks. All memory allocated on the heap must be properly freed before the program ends.

You may use tools like leaks, valgrind, or e_fence to check for memory leaks. If memory leaks are found, tick the appropriate flag.

Attachments

Please download the attachments below:

subject.pdf

cpp_09.tgz

input.csv

Mandatory Part

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) are NOT expected.

The purpose of this module is to use the STL. Then, using the containers is authorized.

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 an external library, or features from versions other than C++98.

Yes No

Exercise 00: Bitcoin Exchange

For this first exercise, you have to find a makefile with the usual compilation rules and the files requested in the subject.

Code review

Check that a makefile is present with the usual compilation rules.

Check in the code that the program uses at least one container.

The person being evaluated must explain why they chose to use this container and not another?

If not, the evaluation stops here.

Yes No

Error handle

Error handle

You must be able to use an empty file or a file with errors (a basic example exists in the subject). The program must not stop its execution before having performed the operations on the whole file passed as argument.

You can use a wrong date.

You can enter a value greater than 1000 or less than 0.

If there is any problem during the execution then the evaluation stops here.

Yes No

Main usage

Main usage

You must now use the "input.csv" file located at the top of this page.

You can modify this file with the values you want.

You have to run the program with the input.csv file as parameter.

Please compare some dates manually with the specified value.

If the date does not exist in the database, the program will have to use the nearest lower date.

Yes No

Exercise 01: Reverse Polish Notation

For this second exercise, you have to find a makefile with the usual compilation rules and the files requested in the subject.

Code review

Check that a makefile is present with the usual compilation rules.

Check in the code that the program uses at least one container.

The person being evaluated must explain why they chose to use this container and not another?

If not, the evaluation stops here.

If the container chosen here is present in the first exercise then the evaluation stops here.

Yes No

Main usage

Main usage

Check that the program runs correctly using different formulas of your choice.

The program is not required to handle expressions with parenthesis or decimals number.

If there is any problem during the execution then the evaluation stops here.

Yes No

Usage advanced

Check that the program runs correctly using different formulas of your choice.

Here is some tests:

```bash

8 9 * 9 9 9 4 1 +

> Result: 42

9 8 * 4 * 4 / 2 + 9 8 8 1 6 -

> Result: 42

1 2 * 2 / 2 + 5 * 6 1 3 * 4 5 * * 8 /
```

> Result: 15

`

You can use the examples in the topic if you don't know which formula to use.

If there is any problem during the execution then the evaluation stops here.

Yes No

Exercise 02: PmergeMe

As usual, there has to be enough tests to prove the program works as expected. If there isn't, do not grade this exercise. If any non-interface class is not in orthodox canonical class form, do not grade this exercise.

Code review

Check that a makefile is included with the usual compilation rules rules.

Check in the code that the program uses at least two containers.

If not, the evaluation stops here.

The person being evaluated must explain why they chose to use these containers and not another?

Check in the code that the merge-insert sort algorithm is present and is used for each container. The Ford-Johnson algorithm must be used.

A brief explanation is expected. In case of doubt, the evaluation stops here.

If one of the containers chosen here is included in one of the previous exercises then the evaluation stops here.

Yes No

Main usage

You can now manually check that the program works correctly by using between 5 and 10 different positive integers of your choice as program arguments.

If this first test works and gives a sorted sequence of numbers you can continue.

If not, the evaluation stops now.

Now you have to check this operation by using the following command as an argument to the program:

For linux:

```bash
shuf -i 1-1000 -n 3000 | tr "\n" " "
```

For OSX:

```bash
jot -r 3000 1 1000 | tr '\n' ' '
```

If the command works correctly, the person being evaluated should be able to explain the difference in time used for each container selected.

If there are any problems during the execution and/or explanation then the evaluation stops here.

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