

You should evaluate 1 student in this team

# Introduction

- 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 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

 $\frac{1}{4}$

☐ subject.pdf (<https://cdn.intra.42.fr/pdf/pdf/146749/en.subject.pdf>)

☐ cpp\_09.tgz ([https://cdn.intra.42.fr/document/document/29461/cpp\\_09.tgz](https://cdn.intra.42.fr/document/document/29461/cpp_09.tgz))

## Preliminary tests

*If cheating is suspected, the evaluation stops here. Use the "Cheat" flag to report it. Take this 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 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

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

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 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

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:

```
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, 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 algorithm must be used.

Therefore, the student must be capable of explaining the following concepts:

- The key aspects of merge insertion, specifically the role of pairs.
- The Jacobsthal sequence and its relevance.
- The process of binary search.

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:

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

For OSX:

```
`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

## Ratings

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

☐ Ok

☐ Outstanding project

☐ Empty work

☐ Incomplete work

☐ Invalid compilation

☐ Cheat

☐ Crash

☐

☐ Leaks

☐ Forbidden function

☐ Can't support / €

## Conclusion

Leave a comment on this evaluation ( 2048 chars max )

Finish evaluation