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

stamim

(https://
profile.intra.42.fr)

SCALE FOR PROJECT OCAML - RECURSION AND HIGHER-ORDER FUNCTIONS - 0 (/PROJECTS/OCAML-RECURSION-AND-HIGHER-ORDER-FUNCTIONS-0)

You should evaluate 1 student in this team



Git repository

git@vogsphere-v2.1337.ma:vogsphere/intra-uuid-996acbb0-f29c-41e0



Introduction


For the sake of this evaluation, we ask you to:

- Stay mannerly, polite, respectful and constructive during this evaluation. The trust between you and the 42 community depends on it.
- Bring out to the graded student (or team) any mistake he or she might have done.
- Accept that there might be differences of interpretation of the subject or the rules between you and the graded student (or team). Stay open-minded and grade as honestly as possible.

Guidelines

- You must grade only what is present in the graded student's (or team) repository. As such, any non-tested feature is a non-functional feature.
- You must stop grading at the first failed exercise, but you are encouraged to continue testing and discussing the following exercises.

Attachments

 subject.pdf (<https://cdn.intra.42.fr/pdf/pdf/144419/en.subject.pdf>)

Preliminaries

This section is dedicated to set the evaluation up and test its prerequisites. It doesn't reward any points, but if something is wrong at this step or at any point of the evaluation, the final grade is 0, and an appropriate flag might be checked if needed.

Respect of the rules

- The graded student (or team)'s work is present on her or his repository.
- The graded student (or team) is able to explain his or her work at any time of the evaluation.
- The general rules and the possible day-specific rules are respected at any time of the evaluation.
- For this project, you need to clone the Git repository on the evaluated person's computer.

☒ Yes

☐ No

OCaml - Recursion and higher-order functions - 0

- For each exercise, you must compile the exercise using `ocamlopt` and run the generated executable. If the compilation fails or throws a warning, or an exception is thrown at runtime, the exercise is failed. - Remember that if the student's tests cannot prove that his exercise is working properly, he cannot earn points for it. - Remember to check function names, types, behaviours and outputs. - More than ONE let-definition (other than the `let` ()) means the exercise is failed. No questions asked.

ex00 repeat_x

The function has to be tested at least with: a negative value, zero, a positive value.

☒ Yes

☐ No

ex01 repeat_string

The following features have to be tested: Negative value, Zero, Positive value without specifying a string, Positive value and specifying a string.

☒ Yes

☐ No

ex02 ackermann

The function has to be tested at least with a negative number

and another case with both positive arguments. For every test, check the results match exactly. (Results table available on Wikipedia).

☒ Yes☐ No

ex03 tak

There must be at least 3 tests. The execution time has to be near-instantaneous.

☒ Yes☐ No

ex04 fibonacci

The function has to be at least tested with: A negative value, A positive value (with results matching the reference on Wikipedia)

The student must be able to prove that his function is tail-recursive and runs in linear time (hint: the execution must be near-instantaneous).

☒ Yes☐ No

ex05 hfs_m & hfs_f

Both functions must be tested with at least a negative value and a positive value. For every test, check that the results match the reference from Wikipedia.

☒ Yes☐ No

ex06 iter

This function must be tested at least once. Check that the results are correct.

☒ Yes☐ No

ex07 converges

This function must be tested with at least a case where it should return true, and a case where it should return false. Of course, all the tested results must be correct.

☒ Yes☐ No

ex08 ft_sum

This function must be tested at least with a normal case and a lower bound greater than the upper bound (for which the function should return nan).

The student must be able to prove that his function is tail-recursive and runs in linear time.

☒ Yes

☐ No
ex09 leibniz_pi

This function must at least be tested with a negative and a positive delta.

The student must be able to prove that his function is tail-recursive and runs in linear time.

☒ Yes

☐ No

Ratings

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

☒ Ok

☐ Empty work

☐ Incomplete work

☐ Invalid compilation

☐ Cheat

☐ Crash

☐ Incomplete group

☐ Concerning situation

☐ Forbidden function

☐ Can't support / explain code

Conclusion

Leave a comment on this evaluation

Finish evaluation

Declaration on the use of cookies (<https://profile.intra.42.fr/legal/terms/2>)

Privacy policy (<https://profile.intra.42.fr/legal/terms/5>)

General term of use of the site (<https://profile.intra.42.fr/legal/terms/6>)

Rules of procedure (<https://profile.intra.42.fr/legal/terms/4>)

Terms of use for video surveillance (<https://profile.intra.42.fr/legal/terms/1>)

Legal notices (<https://profile.intra.42.fr/legal/terms/3>)