

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



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

ylabtaim

SCALE FOR PROJECT OCAML - MONOIDS AND MONADS - 3 (/PROJECTS/OCAML-MONOIDS-AND-MONADS-3)

You should evaluate 1 student in this team



Git repository

`git@vogsphere-v2.1337.ma:vogsphere/intra-uuid-c7f4388a-cdbd-4673`



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.
- For each exercise, you must compile the exercise using `ocamlopt` and run the generated executable. If the compilation fails or warns, 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.

Attachments

 subject.pdf (<https://cdn.intra.42.fr/pdf/pdf/146473/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 in his or her 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.

☒ Yes

☐ No

Ex00, All Along the Watchtower!

Test the Watchtower monoid. It should contain :

- the zero should be 12
- the add and the sub rules must add two hours and use mod 12 to avoid getting out of the type hour ($3h + 14h = 17h \bmod 12h = 5h$) also the sub rule must not return a negative number!
- the zero should be 0
- the add and the sub rules must add two hours and use mod 12 to avoid getting out of the type hour ($3h + 14h = 17h \bmod 12h = 5h$)

ALL tests MUST be implemented by the student. Again, if something is missing, the feature won't be graded.

☒ Yes

☐ No

Ex01, The "Alan Parson's Project"

Test the Project monoid. It should contain :

- a project type as an alias of `string * string * int`
- a zero which is `("", "", 0)`

- a combine rule that concatenate the first strings, average of ints as int and a status relativ to this average value.
- a fail rule that creates a new project by setting the status to failed.
- a success rule that creates a new project by setting the status to succeed and the grade to 80.

ALL tests MUST be implemented by the student. Again, if something is missing, the feature won't be graded.

✓ Yes

✗ No

Ex02, These aren't the functoids you're looking for

Test the INT and FLOAT monoids. they should contain :

- a type named element that is an alias of int for INT and an alias of float for FLOAT
- a zero1 for add and sub (0 and 0.0 for INT and FLOAT)
- a zero2 for mul and div (1 and 1.0 for INT and FLOAT)
- 4 rules : add, sub, mul and div implemented (2 pts)

Test the Calc functor that should implement :

- all 4 rules : add, sub, muland div by using element from the Monoid M as parameters and rules form the Monoid M as rules.
- a power function that calculate the power of an M.element by the int power. (Be carefull power x 0 should return M.zero2!)
- a fact function that caculate the factorial of a M.element. Again be carefull, fact M.zero1 and fact M.zero2 should return M.zero2 (3 pts)

ALL tests MUST be implemented by the student for EACH RULE.
Again, if something is missing, the feature won't be graded.

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



Ex03, Try

This monad is simple. It should implement the following functions, with tests to prove their correct behaviour:

- return
- bind
- recover
- filter
- flatten

Don't forget the student must be able to explain his code (and also explain what a monad is).

✓ Yes

✗ No

Ex04, Set

This monad should implement the following functions, with tests to prove their correct behaviour:

- return
- bind
- union
- inter
- diff
- filter
- foreach
- for_all
- exists

Don't forget the student must be able to explain his code (and also explain what a monad is).

☒ 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>)

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

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

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

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>)