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

SCALE FOR PROJECT PISCINE OCAML (/PROJECTS /42CURSUS-PISCINE-OCAML) / DAY 08 (/PROJECTS /42CURSUS-PISCINE-OCAML-DAY-08)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere.msk.21-school.ru:vogsphere/intra-uuid-65f74102-5a1d-48⌐  📋
```

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/27343/en.subject.pdf)

Preliminaries

This section is dedicated to set the evaluation up and test its prerequirements. 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 approriate 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 Atoms

Mandatory points

This class is really straightforward, but all the fields have to be there. The to_string method has to include all three fields (name, symbol, atomic_number) in its result, and the class has to be virtual. There must be six atoms in total, including hydrogen, carbon and oxygen (with correct symbols and atomic numbers).

    ☑ Yes   ✕ No

Bonus points

How many more atoms are implemented? If you think some of them are unusual or funny, you may give the student two points for each of those.

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

◯

ex01 Molecules

Mandatory points

This class is also really simple, but there are some requirements. Its constructor should only accept a name and a list of atoms, and the formula should be returned formatted with

the Hill notation. The to_string method should include at least
the name and the formula in its result, and the the equals
method should return true for two instances of the same molecule
(it is okay if that particular feature is not tested, but you
should check that the equals method works like that). The
molecule class must be virtual. There should be at least five
concrete molecules implemented along with the molecule class,
including water and carbon dioxyde.

⊘ Yes   ✕ No

Bonus points

How many more molecules are implemented? If they are complex or
unusual, you can give the student two points for each molecule.
Other molecules earn one point each.

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

ex02 Alkanes

Mandatory points

An alkane should provide at least the same methods as a molecule
(name, formula, to_string, equals), and its constructor must
only accept an argument n. The name and formula have to be
correct, and the formula must obey to the Hill notation.
Obviously, incorrect values of n must be handled correctly (with
an exception for instance, but anything is fine as long as it
makes sense). There must also be at least methane (1), ethane
(2) and octane (8) implemented.

⊘ Yes   ✕ No

ex03 Reactions

Mandatory points

All the methods specified in the exercise (get_start,
get_result, balance, is_balanced) have to be present with the
right types (these methods are all virtual). As a consequence,
there is nothing really to test, but if the student made a dummy
reaction class and tested it that is really a great idea.

⊘ Yes   ✕ No

ex04 Alkane combustion

Mandatory points

This is where it gets hard. There has to be a class named alkane_combustion, and there are basically two states. First you instantiate the class with a list of alkanes; but that combustion is not balanced yet, so get_start and get_result throw an exception, and is_balanced return false. Then you call balance on it, which returns a new object. This combustion has to be balanced, is_balanced has to return true, and get_start and get_result have to return consistent results (they have to make up a really balanced alkane combustion.)

    ⊘ Yes   ✕ No

ex05 Incomplete combustions

Mandatory points

This is a new method on the alkane_combustion class. It has to be named get_incomplete_results, and return an associative list. The index will be the quantity of molecular oxygen at the start of the combustion, and the value will be the different possible outcomes. Check that all the results are valid, without duplicates, and comprehensive (if you can find a solution which is not included in the function's returned value, the exercise is obviously incorrect).

    ⊘ Yes   ✕ No

Ratings

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

✔ Ok

📄 Empty work   💬 No author file   🌐 Invalid compilation   🎵 Norme   📑 Cheat   ☢ Crash

⊘ Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation