

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

# SCALE FOR PROJECT OCAML - OCAML'S MODULES LANGUAGE - 1 (/PROJECTS/OCAML- OCAML-S-MODULES-LANGUAGE-1)

You should evaluate 1 student in this team



Git repository

git@vogsphere-v2.1337.ma:vogsphere/intra-uuid-30f58e68-de7e-4b5a



## Introduction

For the good 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 she or he might did.
- 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 and the graded student's (or team) repository.
- 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/144421/en.subject.pdf)

# Preliminaries

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

## Respect of the rules

- The graded student (or team) work is present on her or his repository.
- The graded student (or team) is able to explain her or his 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's modules language - 1

*- For each exercice, you must compile the exercice using `ocamlopt` and run the generated executable. If the compilation fails or warns, or an unexpected exception is thrown at runtime, the exercice is failed. - Whether the graded student provided tests or not, you must test her or his work extensively and asses if the work is done or not. - Remember to check function names, types, behaviours and outputs.*

## Ex00, cards colors

To test this exercice, copy the following code to a file named "Color.mli" and compile using the command "`ocamlopt Color.mli Color.ml main.ml`".

```
$cat Color.mli type t = Spade | Heart | Diamond | Club val all :
t list val toString : t -> string val toStringVerbose : t ->
string $
```

Run the executable and check that the value "all" and the functions "toString" and "toStringVerbose" behave as stated in the subject.

☒ Yes

☐ No

## Ex01, cards values

To test this exercice, copy the following code to a file named "Value.mli" and compile using the command "`ocamlopt Value.mli Value.ml main.ml`".

```
$cat Value.mli type t = T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 |
T10 | Jack | Queen | King | As val all : t list val toInt : t ->
```

```
int val toString : t -> string val toStringVerbose : t -> string
val next : t -> t val previous : t -> t $
```

Run the executable and check that the value "all" and the 5 functions behave as stated in the subject, including the error cases for the functions "previous" and "next".

☒ Yes

☐ No

### Ex02, cards

To test this exercise, copy the following code to a file named "Card.mli" and compile using the command "ocamlc Card.mli Card.ml main.ml".

```
$cat Card.mli module Color : sig type t = Spade | Heart |
Diamond | Club val all : t list val toString : t -> string val
toStringVerbose : t -> string end module Value : sig type t = T2
| T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | Jack | Queen | King |
As val all : t list val toInt : t -> int val toString : t ->
string val toStringVerbose : t -> string val next : t -> t val
previous : t -> t end type t val newCard : Value.t -> Color.t ->
t val allSpades : t list val allHearts : t list val allDiamonds
: t list val allClubs : t list val all : t list val getValue : t
-> Value.t val getColor : t -> Color.t val toString : t ->
string val toStringVerbose : t -> string val compare : t -> t ->
int val max : t -> t -> t val min : t -> t -> t val best : t
list -> t val isOf : t -> Color.t -> bool val isSpade : t ->
bool val isHeart : t -> bool val isDiamond : t -> bool val
isClub : t -> bool $
```

Run the executable and check that every values and functions from Card, Card.Color and Card.Value behave as stated in the subject, including the error cases for the functions that might fail on specific values.

☒ Yes

☐ No

### Ex03, decks

Check that the file "Deck.mli" provides an interface for the module "Deck" that respects the following statements:

- The Deck module embeds the Card module from the previous exercise.
- The Deck module exposes an abstract type t that represents a deck. Its definition is up to you.
- The Deck module exposes a function "newDeck" of type "unit -> t".
- The Deck module exposes a function "toStringList" of type "t

-> string list".

- The Deck module exposes a function "toStringListVerbose" of type "t -> string list".
- The Deck module exposes a function "drawCard" of type "t -> Card.t \* t", or "t -> (Card.t \* t)", both types are equivalent.

Compile using the command "ocamlopt Deck.mli Deck.ml main.ml".

Run the executable and check that the following statements are true:

- The modules Deck.Card, Deck.Card.Color and Deck.Card.Value behave as stated in the subject.
- The function newDeck returns a shuffled deck. Two successive calls to this function must not return the same order of cards.
- The function toString returns a list of strings of each card remaning in the deck.
- The function toStringVerbose returns a list of strings of each card remaning in the deck in verbose formating.
- The function drawCard returns a couple composed of the first card of the deck and the rest of the deck.
- 52 successive calls to the function drawCard on the same deck raises the "Failure" exception with a relevant error message.

✓ 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](https://profile.intra.42.fr/legal/terms/2))

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

Legal notices  
([https://  
profile.intra.42.fr/  
legal/terms/3](https://profile.intra.42.fr/legal/terms/3))

Privacy policy  
([https://  
profile.intra.42.fr/  
legal/terms/5](https://profile.intra.42.fr/legal/terms/5))

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

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