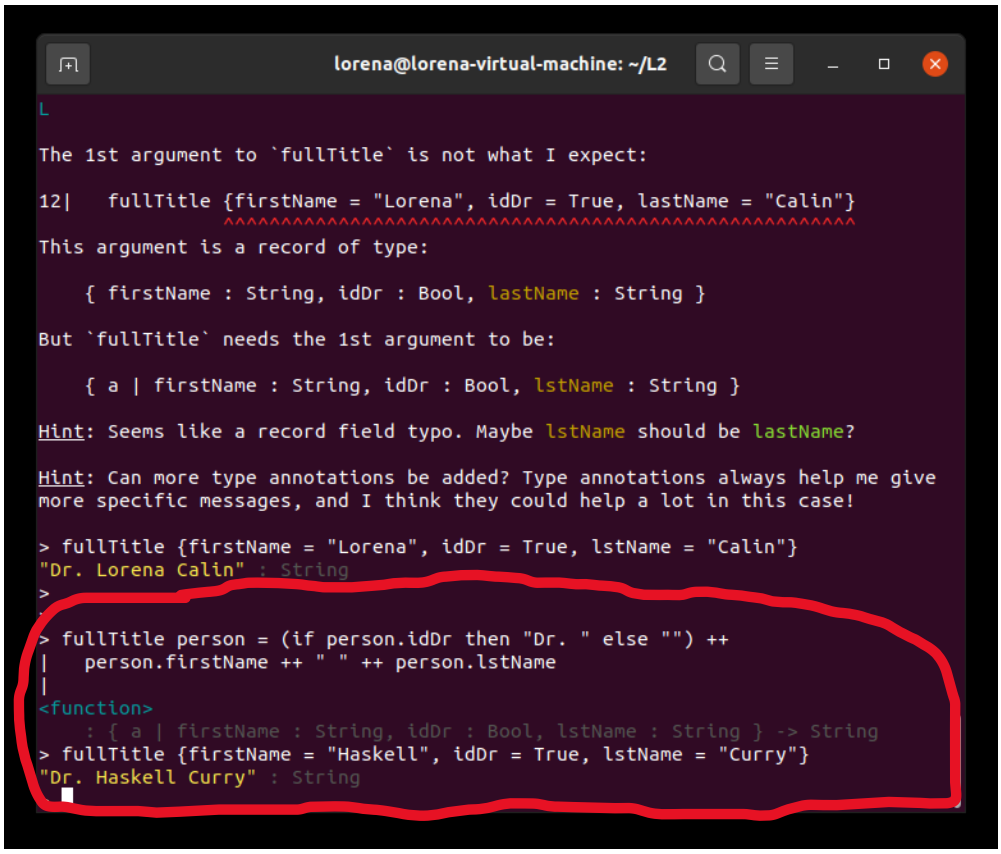
LABORATORY 2

Exercise 2.2.1:
fullTitle person = (if person.idDr then "Dr. " else "") ++ person.firstName ++ " " ++ person.lstName
Try to call the function with an argument such that "Dr. Haskell Curry" is displayed.



Exercise 2.3.1: Call the fullName function using the User type constructor. Did you encounter any errors?
ANSWER: I encountered a lot of error before managing to write correctly… You can see below some of them:

lorena@lorena-virtual-machine: ~/L2

```
>
> fullName User
-- TYPE MISMATCH ------------------------------------------------------------ RE
PL

The 1st argument to `fullName` is not what I expect:

13|   fullName User
              ^^^^
This `User` value is a:

    String -> String -> User

But `fullName` needs the 1st argument to be:

    User

> fullName User "Lorena" "C"
-- TYPE MISMATCH ------------------------------------------------------------ RE
PL

The 1st argument to `fullName` is not what I expect:

13|   fullName User "Lorena" "C"
              ^^^^
This `User` value is a:

    String -> String -> User

But `fullName` needs the 1st argument to be:

    User

-- TOO MANY ARGS ------------------------------------------------------------ RE
PL

The `fullName` function expects 1 argument, but it got 3 instead.

13|   fullName User "Lorena" "C"
      ^^^^^^^^
Are there any missing commas? Or missing parentheses?
```

lorena@lorena-virtual-machine: ~/L2

```
13|   fullName User("Lorena", "Calin")
      ^^^^^^^^
Are there any missing commas? Or missing parentheses?

> fullName User("Lorena" "C")
-- TOO MANY ARGS ------------------------------------------------------------ RE
PL

This value is not a function, but it was given 1 argument.

13|   fullName User("Lorena" "C")
                   ^^^^^^^^
Are there any missing commas? Or missing parentheses?

-- TYPE MISMATCH ------------------------------------------------------------ RE
PL

The 1st argument to `fullName` is not what I expect:

13|   fullName User("Lorena" "C")
              ^^^^
This `User` value is a:

    String -> String -> User

But `fullName` needs the 1st argument to be:

    User

-- TOO MANY ARGS ------------------------------------------------------------ RE
PL

The `fullName` function expects 1 argument, but it got 2 instead.

13|   fullName User("Lorena" "C")
      ^^^^^^^^
Are there any missing commas? Or missing parentheses?

> fullName (User "Lorena" "C")
"Lorena C" : String
>
```

Question 2.3.1: Does the way type alias works remind you of any keyword in C and C++?
ANSWER: This reminds me of typedef keyword which was also used to give a type a new name.

Exercise 2.3.2: Define a type alias Address, which includes 4 fields: street, number, city and country.
ANSWER: type alias Address = {street: String, number: Int, city: String, country: String}

Exercise 2.3.3: Write a function formatAddress , which takes an instance of an Address and displays it as street number, city, country.

```
> formatAddress : Address -> String
| formatAddress add = add.street ++ " " ++ String.fromInt(add.number) ++ ", " ++ add.cit
y ++ ", " ++ add.country
|
<function> : Address -> String
> formatAddress (Address "Baritiu street" 26 "Cluj-Napoca" "Romania")
"Baritiu street 26, Cluj-Napoca, Romania" : String
>
>
```

Exercise 2.5.1: Try to remove the last line (_ -> "Better luck next time") and check if the code could be compiled.

```
lorena@lorena-virtual-machine: ~/L2
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
> numberToMedal : Int -> String
| numberToMedal n =
|   case n of
|   1->"Gold"
|   2->"Silver"
|   3->"Bronze"
|
-- MISSING PATTERNS ---------------------------------------------------- REPL

This `case` does not have branches for all possibilities:

12|>  case n of
13|>  1->"Gold"
14|>  2->"Silver"
15|>  3->"Bronze"

Missing possibilities include:

    _

I would have to crash if I saw one of those. Add branches for them!

Hint: If you want to write the code for each branch later, use `Debug.todo` as a
placeholder. Read <https://elm-lang.org/0.19.1/missing-patterns> for more
guidance on this workflow.
```

ANSWER: We can't compile without having a catch-all case in the case expression. ☹

Exercise 2.5.2: Try to swap the 1 -> "Gold" and _ -> "Better luck next time" lines. Evaluate the following expressions in the REPL (numberToMedal 1), (numberToMedal 2), (numberToMedal 10).

```
lorena@lorena-virtual-machine: ~/L2

> :reset
<reset>
> numberToMedal : Int -> String
| numberToMedal n =
|   case n of
|   _ -> "Better luck next time"
|   2 -> "Silver"
|   3 -> "Bronze"
|   1 -> "Gold"
|
-- REDUNDANT PATTERN ------------------------------------------------------- REPL

The 2nd pattern is redundant:

4|   case n of
5|   _ -> "Better luck next time"
6|>  2 -> "Silver"
7|   3 -> "Bronze"
8|   1 -> "Gold"

Any value with this shape will be handled by a previous pattern, so it should be
removed.

> numberToMedal 1
-- NAMING ERROR ------------------------------------------------------------ REPL

I cannot find a `numberToMedal` variable:

3|   numberToMedal 1
     ^^^^^^^^^^^^^
These names seem close though:

    Maybe.map
    Sub.map
    Sub.none
    Char.toCode

Hint: Read <https://elm-lang.org/0.19.1/imports> to see how `import`
declarations work in Elm.

>
```

Each time I tried to run this module I got this error and, honestly, I can't see where I'm wrong… Or maybe it's the fact that the "_" case cannot be the first one, because it works well in the initial form.

Question 2.7.1: What is the cardinality of the Bool type?
ANSWER: The Bool type has 2 variants: True and False, so its cardinality is 2.

Exercise 2.8.1: Define a type for a dice which has six sides.

```
> :reset
<reset>
> type DiceFace = One | Two | Three | Four | Five | Six
> Six
Six : DiceFace
>
```

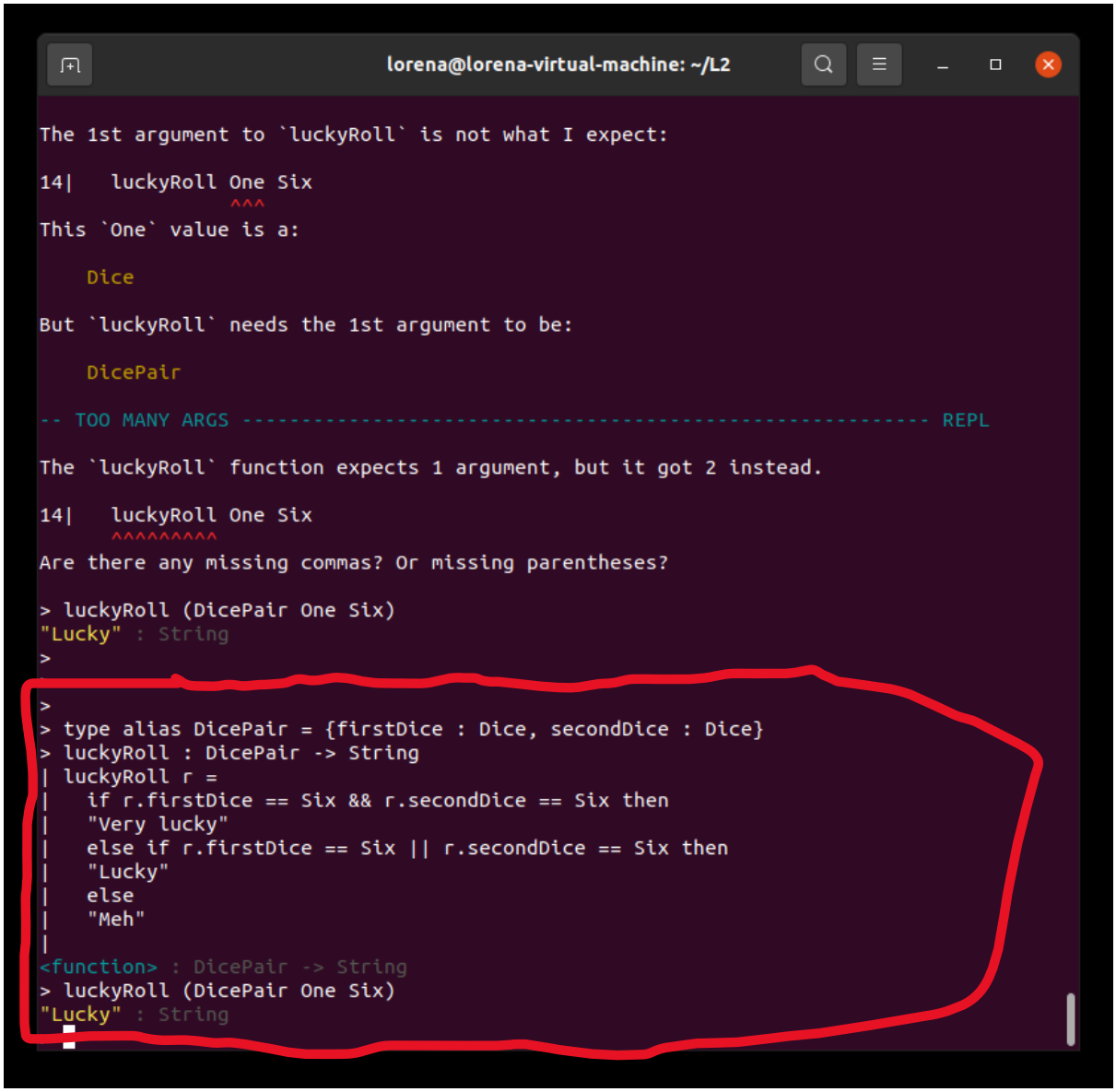
Exercise 2.8.2: Define a type DicePair , which contains 2 Dice , in two ways, one using type aliases and one using type definitions.

> type Dice = One | Two | Three | Four | Five | Six

> type alias DicePair = {firstDice : Dice, secondDice : Dice}

> type DicePair = DicePair Dice Dice

Exercise 2.8.3: Write a function luckyRoll which takes a DicePair and returns a String. It should return “Very lucky” if the roll contains 2 sixes, “Lucky” it contains one six and “Meh” otherwise.



Exercise 2.8.4: Write the function areaRec for ShapeRec

I used the function which implements Heron's formula for computing the area of a triangle knowing the edges from the laboratory notes.
Then I used case expression for computing the area depending on the type of shape: circle, rectangle or triangle.

```
lorena@lorena-virtual-machine: ~/L2

    TriangleRec _

I would have to crash if I saw one of those. Add branches for them!

Hint: If you want to write the code for each branch later, use `Debug.todo` as a
placeholder. Read <https://elm-lang.org/0.19.1/missing-patterns> for more
guidance on this workflow.

> heronShort a b c =
|   let s = (a+b+c)/2
|   in
|   sqrt (s * (s-a) * (s-b) * (s-c))
|
<function> : Float -> Float -> Float -> Float
>
> heronShort 2 2 3
1.984313483298443 : Float
>
> type ShapeRec = CircleRec {radius : Float} | RectangleRec {width : Float, height : F
loat} | TriangleRec {sideA : Float, sideB : Float, sideC : Float}
> areaRec : ShapeRec -> Float
| areaRec shapeRec =
|   case shapeRec of
|   CircleRec {radius} -> pi * radius * radius
|   RectangleRec {width, height} -> width * height
|   TriangleRec {sideA, sideB, sideC} -> heronShort sideA sideB sideC
|
<function> : ShapeRec -> Float
>
> areaRec (CircleRec {radius = 2})
12.566370614359172 : Float
> areaRec (TriangleRec {sideA = 3, sideB = 4, sideC = 5})
6 : Float
>
>
```

Exercise 2.8.6: Write a function validateCard : Date -> CreditCard -> Bool which checks if a credit card is valid.

1. Define the Date type, which stores the month and year until a card is valid.
2. Define the CardNumber type, which stores the 16 digits of the card as 2 Int s of 8 digits each. This is necessary because a 16 digit, positive integer can't be stored in a 32 bit Int type.
3. Define the CreditCard type for a credit card which has: ˆ an issuer (Visa or Mastercard) ˆ a card number, which is of type CardNumber ˆ an expiration date, which is of type Date 32
4. Write a function isDateAfter to check if the second date is after the first date.
5. Write a function isCardNumberValid to check if the credit card number is valid:

(a) To check that the whole number is valid, use the Luhn algorithm.

(b) If the INN (Issuer Identification Number) matches the card issuer: ˆ Visa cards start with the digit 4 ˆ Mastercard cards have the first 4 digits between 2221 and 2720 or have the first 2 digits between 51 and 55

ANSWER: In the first screenshot there are the type declarations, as required in exercises 1,2,3.

```
lorena@lorena-virtual-machine: ~/L2
>
>
>
> type alias Date = {month: Int, year: Int}
> type alias CardNumber = {firstEight: Int, secondEight: Int}
> type Issuer = Visa | Mastercard
> type alias CreditCard = {issuer: Issuer, cardNumber: CardNumber, expirationDate: Da
te}
>
```

In the second picture there is the function isDateAfter required in exercise 4.

```
lorena@lorena-virtual-machine: ~/L2
12|   else if date2.year == date1.year && date2.month < date1.month then
13|   True
14|   else if date2.year > date1.year then
                                         ^
I was expecting to see an expression next. Maybe it is not filled in yet?

Note: I can be confused by indentation, so if the `then` branch is already
present, it may not be indented enough for me to recognize it.

>
> isDateAfter: Date -> Date -> Bool
| isDateAfter date1 date2 =
|   if date2.year == date1.year && date2.month > date1.month then
|   True
|   else if date2.year == date1.year && date2.month < date1.month then
|   False
|   else if date2.year > date1.year then
|   True
|   else
|   False
|
<function> : Date -> Date -> Bool
> isDateAfter (Date 4 2002) (Date 5 2002)
True : Bool
> isDateAfter (Date 4 2002) (Date 2 2002)
False : Bool
> isDateAfter (Date 4 2002) (Date 2 2005)
True : Bool
> isDateAfter (Date 4 2002) (Date 4 2005)
True : Bool
> isDateAfter (Date 4 2002) (Date 4 2000)
False : Bool
>
>
>
```