Assignment 1

Basic concepts

Deadline: Monday, October 12, 13:55

1.1 Submission instructions

1. Unzip the `A1.zip` folder.
2. Edit the first line of the `Solutions.Elm` file as described in the comments.
3. Edit the `Solutions.Elm` file in the `src` folder with your solutions.
4. When done, zip this `A1` folder and name the zip archive with the following format:

$$A1\_\langle FirstName\rangle\_\langle LastName\rangle\_\langle Group\rangle$$

Examples of valid names:

- `A1_John_Doe_30432.zip`
- `A1_Ion_Popescu_30434.zip`
- `A1_Gigel-Dorel_Petrescu_30431.zip`

Examples of invalid names:

- `Solutions.zip`
- `A1.zip`
- `Solutii_A1_Ion_Popescu.zip`

1.2 Assignment exercises

Exercise 1.2.1 3p

Define the types `Face`, `Suit` and `Card` to represent the cards of a standard 52-card deck[a].

Grading:
1 point for each correct definition.

[a] `https://en.wikipedia.org/wiki/Standard_52-card_deck`

Exercise 1.2.2 2p

Define functions for each of the `Face`, `Suit` and `Card` type to print their value as a string. For the `Card` type it should print it in the format "⟨*Face*⟩ *of* ⟨*Suit*⟩" (e.g. `"Ace of Spades"`).

Grading:

- 0.5 points for the correct definition of `faceToString`
- 0.5 points for the correct definition of `suitToString`
- 1 point for the correct definition of `cardToString`

Exercise 1.2.3 2p

Write an Elm function `linesIntersect`, which given two line segments defined by their endpoints, computes whether the two segments intersect or not.

Grading:

- 1 point for implementing a function which returns correct results
- 1 point for defining suitable data types for the problem

Exercise 1.2.4 3p

Write an Elm function called `trailingZeroes n`, which computes the number of trailing zeros in n factorial.

Elm REPL

```
> trailingZeroes 5
1 : number
> trailingZeroes 10
2 : number
> trailingZeroes 42
9 : number
> trailingZeroes 100
24 : number
> trailingZeroes 1000
249 : number
```

Grading:

- 1 point for implementing a function which returns correct results
- 1 point for correctly using local declarations (`let ... in`)
- 1 point for correctly using tail recursion