Learning to Program with F#
Exercises
Department of Computer Science
University of Copenhagen

Jon Sporring, Martin Elsman, Torben Mogensen, Christina Lioma

October 21, 2022

0.1 Exceptions

0.1.1 Teacher's guide

Emne Untagelser og option typen

Sværhedsgrad Let

0.1.2 Introduction

Denne opgave omhandler undtagelser (exceptions), option typer og Stirlings formel. Stirlings formel er en approximation til fakultetsfunktionen via

$$\ln n! \simeq n \ln n - n.$$

0.1.3 Exercise(s)

0.1.3.1: Implement the faculty function $n! = \prod_{i=1}^{n} i$, $n > 0$ as `fac : n:int -> int`. The function must cast a `System.ArgumentException` exception, if the function is called with $n < 1$. Call `fac` with the values $n = -4, 0, 1, 4$, and catch possible exceptions.

0.1.3.2: Add a new and selfdefined exception `ArgumentTooBig of string` to `fac` in Assignment 1, and cast it with the argument `"calculation would result in an overflow"`, when n is too large for the `int` type. Call the function with a small and a large value of n, catch the possible exception and handle it in case by writing the exception message to the screen.

0.1.3.3: Make a new faculty function `facFailwith : n:int -> int`, as `fac` in Assignment 2, but where the 2 exceptions are replaced with `failwith` with the arguments `"argument must be greater than 0"` and `"calculation would result in an overflow"` respectively. Call `facFailWith` with $n = -4, 0, 1, 4$, catch possible exceptions with the `Failure` pattern, and write the returned message from `failwith` to the screen.

0.1.3.4: Write a new faculty function as in Assignment 2 but with the name and type `facOption : n:int -> int option`, which returns `Some m`, when the result is computable and `None` otherwise. Call `fac` with the values $n = -4, 0, 1, 4$, and write the result to the screen.

0.1.3.5: Write a function `logIntOption : n:int -> float option`, which calculates the logarithm of `n`, if `n` > 0 and `None` otherwise. Test `logIntOption` for the values `-10`, `0`, `1`, `10`.

0.1.3.6: Write a function `logFac : int -> float option` which calculates $\log(n!)$ by combining `logIntOption` and `facOption` and using `Option.bind`. Compare the output of `logFac` with Stirlings approximation $\log(n!) \simeq n \log n - n \log e$, where $e = 2.718281\ldots$ is the natural exponential base and for $n = 1, 2, 4, 8$.

0.1.3.7: The function `logFac : int -> float option` can be defined in many ways as a combination of the functions `Some`, `Option.bind`, `logIntOption`, and `facOption`. Write 3 single-line statements, which uses `|>`, `>>` or none of them.

0.1.3.8: Make implementations of the following functions:

```
safeIndexIf : arr:'a [] -> i:int -> 'a
safeIndexTry : arr:'a [] -> i:int -> 'a
safeIndexOption : arr:'a [] -> i:int -> 'a option
```

Each of them must return the value of `arr` at index `i`, when `i` is a valid index, and otherwise handle the error-situation. The error-situations must be handled in different ways:

- `safeIndexIf` must not make use of `try-with` and must not cast an exception.
- `safeIndexTry` must use `try-with`, and it must call `failwith` when there is an error.
- `safeIndexOption` must return `None` in case of an error.

Make a short test of all 3 functions, by writing the content of an array to the screen (and not as an option type). The tests must also include examples of error situations and must be able to handle possible exceptions casted. In your opinion, is any of the above method superior or inferior in how they handle errors and why?