

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

Jon Sparring, Martin Elsmann, Torben Mogensen, Christina Lioma

August 24, 2020

0.1 Exceptions

- 0.1.1:** Implementer fakultetsfunktionen $n! = \prod_{i=1}^n i$, $n > 0$ som `fac : n:int -> int` og kast en `System.ArgumentException` undtagelse, hvis funktionen bliver kaldt med $n < 1$. Kald `fac` med værdierne $n = -4, 0, 1, 4$, og fang evt. undtagelser.
- 0.1.2:** Tilføj en ny og selvdefineret undtagelse `ArgumentTooBig` af `string` til `fac`, og kast den med argumentet `"calculation would result in an overflow"`, når n er for stor til `int` typen. Fang undtagelsen og udskriv beskeden sendt med undtagelsen på skærmen.
- 0.1.3:** Lav en ny fakultetsfunktion `facFailwith : n:int -> int`, som `fac`, men hvor de 2 undtagelser bliver erstattet med `failwith` med hhv. argument `"argument must be greater than 0"` og `"calculation would result in an overflow"`. Kald `facFailWith` med $n = -4, 0, 1, 4$, fang evt. undtagelser vha. `Failure` mønsteret, og udskriv beskeden sendt med `failwith` undtagelsen.
- 0.1.4:** Omskriv fakultetsfunktionen i Opgave 2, som `facOption : n:int -> int option`, således at den returnerer `Some m`, hvis resultatet kan beregnes og `None` ellers. Kald `fac` med værdierne $n = -4, 0, 1, 4$, og skriv resultatet ud vha. en af `printf` funktionerne.
- 0.1.5:** Skriv en funktion `logIntOption : n:int -> float option`, som udregner logaritmen af n , hvis $n > 0$ og `None` ellers. Afprøv `logIntOption` for værdierne $-10, 0, 1, 10$.
- 0.1.6:** Skriv en ny funktion `logFac : int -> float option` vha. `Option.bind` 1 eller flere gange til at sammensætte `logIntOption` og `facOption`, og sammenlign `logFac` med Stirlings approximation $n * (\log n) - n$ for værdierne $n = 1, 2, 4, 8$.
- 0.1.7:** Funktionen `logFac : int -> float option` kan defineres som en enkelt sammensætning af funktionerne `Some` og `Option.bind` en eller flere gange og med `logIntOption` og `facOption` som argument til `Option.bind`. Opskriv 3 udtryk, der bruger hhv. `|>` eller `>>` operatorerne eller ingen af dem.
- 0.1.8:** 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?