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 Expression trees

0.1.1 Teacher's guide

0.1.2 Introduction

The following exercises are about expanding and using the following recursive sumtype, which can be used for modelling expression terms:

```
type expr = Const of int | Add of expr * expr | Mul of expr * expr
```

0.1.3 Exercise(s)

0.1.3.1: Implement a recursive function `eval : expr -> int` that takes an expression value as argument and returns the integer resulting from evaluating the expression term. The expression `eval (Add(Const 3,Mul(Const 2,Const 4)))` should return the integer value `11`.

0.1.3.2: Extend the type `expr` with a case for subtraction, extend the evaluator with a proper `match`-case for subtraction, and evaluate that your implementation works in practice.

0.1.3.3: Extend the type `expr` with a case for division and refine the evaluator function `eval` to have type `expr->(int,string)result`. Evaluate that your implementation will propagate "Divide by zero" errors to the toplevel.