


- Introduction and overview
- Basic types, definitions and functions
- Basic data structures
- ▼ **More advanced data structures**

## Table of Contents

## Tagged values

Week 3 Échéance le déc 12, 2016 at 23:30 UTC 

## Recursive types

Week 3 Échéance le déc 12, 2016 at 23:30 UTC 


## Tree-like values

Week 3 Échéance le déc 12, 2016 at 23:30 UTC 


## Case study: a story teller

Week 3 Échéance le déc 12, 2016 at 23:30 UTC 

## Polymorphic algebraic datatypes

Week 3 Échéance le déc 12, 2016 at 23:30 UTC 

## Advanced topics

Week 3 Échéance le déc 12, 2016 at 23:30 UTC 

- Higher order functions
- Exceptions, input/output and imperative constructs
- Modules and data abstraction

## SYMBOLIC MANIPULATION OF ARITHMETIC EXPRESSIONS (44/44 points)

Abstract syntax trees are a convenient way to represent a syntactic expression in a structured way.

Let us consider arithmetic expressions formed by the following rules:

1. an integer is an arithmetic expression ;
2. if `lhs` and `rhs` are arithmetic expressions then `lhs + rhs` is an arithmetic expression;
3. if `lhs` and `rhs` are arithmetic expressions then `lhs * rhs` is an arithmetic expression.

Such an expression can be represented by a value of type `exp` as defined in the given prelude (as well as the definition of `1 + 2 * 3` as an example).

1. Write the expression `2 * 2 + 3 * 3` in a variable `my_example`.
2. Write a function `eval : exp -> int` that computes the value of an arithmetic expression. The evaluation rules are:
  1. If the expression is an integer `x`, the evaluation is `x`.
  2. If the expression is `lhs + rhs` and `lhs` evaluates to `x` and `rhs` evaluates to `y`, then the evaluation is `x + y`.
  3. If the expression is `lhs * rhs` and `lhs` evaluates to `x` and `rhs` evaluates to `y`, then the evaluation is `x * y`.
3. If an expression is of the form `a * b + a * c` then `a * (b + c)` is a factorized equivalent expression.  
Write a function `factorize : exp -> exp` that implements this transformation on its input `exp` if it has the shape `a * b + a * c` or does nothing otherwise.
4. Write the reverse transformation of `factorize`, `expand : exp -> exp`, which turns an expression of the shape `a * (b + c)` into `a * b + a * c`.
5. Implement a function `simplify: exp -> exp` which takes an expression `e` and:
  1. If `e` is of the shape `e * 0` or `0 * e`, returns the expression `0`.
  2. If `e` is of the shape `e * 1` or `1 * e`, returns the expression `e`.
  3. If `e` is of the shape `e + 0` or `0 + e`, returns the expression `e`.
 and does nothing otherwise.

## Remarks:

1. The symbols (`a`, `b`, `c` and `e`) can match any expressions, not just integers.
2. these are a syntactical rewritings, so two expressions are considered equal if and only if they are exactly the same expressions (simply use the `=` operator to check that).
3. The rewritings have to be done on the first level of the expression only, not recursively and not deeper in the expression. If the toplevel expression does not match the expected pattern, simply return the expression untouched.

## THE GIVEN PRELUDE



Rechercher un cours



```
| EMul of exp * exp
```

```
let example =  
  EAdd (EInt 1, EMul (EInt 2, EInt 3))
```

## YOUR OCAML ENVIRONMENT

```
1 let my_example =  
2   EAdd (EMul (EInt 2, EInt 2), EMul (EInt 3, EInt 3))  
3 ;;  
4  
5 let rec eval e = match e with  
6 | EInt n -> n  
7 | EAdd (a, b) -> eval a + eval b  
8 | EMul (a, b) -> eval a * eval b  
9 ;;  
10  
11 let factorize e = match e with  
12 | EAdd (EMul (a, b), EMul (a', c)) ->  
13   if a = a' then  
14     EMul (a, EAdd (b, c)) else  
15     EAdd (EMul (a, b), EMul (a', c))  
16 | e -> e  
17 ;;  
18  
19 let expand e = match e with  
20 | EMul (a, EAdd (b, c)) -> EAdd (EMul (a, b), EMul (a, c))  
21 | e -> e  
22 ;;  
23  
24 let simplify e = match e with  
25 | EMul (a, EInt 0) -> EInt 0  
26 | EMul (EInt 0, a) -> EInt 0  
27 | EMul (a, EInt 1) -> a  
28 | EMul (EInt 1, a) -> a  
29 | EAdd (a, EInt 0) -> a  
30 | EAdd (EInt 0, a) -> a  
31 | e -> e  
32  
33 ;;
```

Evaluate >

Switch >>

Typecheck

Reset Templ

Full-screen |

Check & Sa

Exercise complete (click for details)

44 pts

Exercise 1: my\_example

Completed, 4 pts

Found my\_example with compatible type.

Correct expression found

4 pts

Exercise 2: eval

Completed, 10 pts

Found eval with compatible type.

Computing eval (EInt (-5))

Correct value -5

1 pt

Computing

eval

```
(EMul (EAdd (EInt 2, EAdd (EAdd (EInt 9, EInt 0), EInt (-3))),  
  EAdd (EMul (EAdd (EInt 1, EInt 0), EInt (-2)), EMul (EInt (-9), EInt 1))))
```

Correct value -88

1 pt

Computing eval (EAdd (EInt 7, EInt (-1)))

Correct value 6

1 pt

Computing eval (EAdd (EInt (-2), EMul (EInt 3, EInt 8)))

Correct value 22

1 pt

Computing eval (EInt (-2))

Correct value -2

1 pt

Computing eval (EInt (-4))

Correct value -4

1 pt

Computing eval (EMul (EInt 4, EMul (EInt (-5), EInt (-5))))

Correct value 100

1 pt

Computing

eval

```
(EAdd  
  (EAdd (EMul (EInt (-10), EInt (-10)),  
    EAdd (EAdd (EInt 3, EInt (-9)), EMul (EInt (-10), EInt (-3)))),  
  EAdd (EInt (-8), EAdd (EInt (-5), EInt (-4))))
```

Correct value 107

1 pt

Computing

eval

```
(EMul  
  (EMul (EMul (EInt 7, EInt 2),  
    EAdd (EMul (EInt (-4), EInt (-10)), EAdd (EInt 4, EInt 3))),  
  EInt (-4)))
```

Correct value -2632

1 pt

Computing

Correct value - 1200

1 pt

## v Exercise 3: factorize

Completed, 10 pts

Found factorize with compatible type.

Computing

factorize

(EAdd

(EMul

(EMul (EMul (EInt 5, EMul (EInt (-5), EMul (EInt 0, EInt 7))),

EAdd (EInt (-9),

EAdd (EMul (EInt 7, EInt (-1)), EMul (EInt 1, EInt 0))),

EMul (EInt (-1), EInt 7)),

EMul

(EMul (EMul (EInt 5, EMul (EInt (-5), EMul (EInt 0, EInt 7))),

EAdd (EInt (-9),

EAdd (EMul (EInt 7, EInt (-1)), EMul (EInt 1, EInt 0))),

EAdd

(EAdd (EMul (EAdd (EInt (-9), EInt (-8)), EAdd (EInt (-6), EInt 6)),

EAdd (EInt (-4), EAdd (EInt (-5), EInt (-6))),

EAdd (EInt (-10), EInt (-4))))))

Correct value

1 pt

(EMul

(EMul (EMul (EInt 5, EMul (EInt (-5), EMul (EInt 0, EInt 7))),

EAdd (EInt (-9), EAdd (EMul (EInt 7, EInt (-1)), EMul (EInt 1, EInt 0))),

EAdd (EMul (EInt (-1), EInt 7),

EAdd

(EAdd (EMul (EAdd (EInt (-9), EInt (-8)), EAdd (EInt (-6), EInt 6)),

EAdd (EInt (-4), EAdd (EInt (-5), EInt (-6))),

EAdd (EInt (-10), EInt (-4))))))

Computing

factorize

(EAdd

(EMul

(EAdd

(EMul (EInt (-9),

EAdd (EAdd (EInt 3, EInt 7), EMul (EInt 6, EInt (-8))))),

EAdd (EAdd (EInt 0, EInt (-1)), EAdd (EInt 4, EInt 7))),

EMul

(EMul

(EMul (EAdd (EInt (-1), EInt (-10)), EMul (EInt (-8), EInt (-3))),

EAdd (EInt (-5), EInt (-6))),

EMul (EAdd (EMul (EInt (-8), EInt (-1)), EMul (EInt (-1), EInt (-1))),

EAdd (EInt 5, EInt 8))))),

EMul

(EAdd

(EMul (EInt (-9),

EAdd (EAdd (EInt 3, EInt 7), EMul (EInt 6, EInt (-8))))),

EAdd (EAdd (EInt 0, EInt (-1)), EAdd (EInt 4, EInt 7))),

EInt 1)))

Correct value

1 pt

(EMul

(EAdd

(EMul (EInt (-9), EAdd (EAdd (EInt 3, EInt 7), EMul (EInt 6, EInt (-8))))),

EAdd (EAdd (EInt 0, EInt (-1)), EAdd (EInt 4, EInt 7))),

EAdd

(EMul

(EMul (EMul (EAdd (EInt (-1), EInt (-10)), EMul (EInt (-8), EInt (-3))),

EAdd (EInt (-5), EInt (-6))),

EMul (EAdd (EMul (EInt (-8), EInt (-1)), EMul (EInt (-1), EInt (-1))),

EAdd (EInt 5, EInt 8))))),

EInt 1)))

Computing

factorize

(EAdd

(EMul

(EMul

(EAdd (EInt (-10),

EAdd (EMul (EInt (-1), EInt 3), EAdd (EInt 1, EInt (-1))))),

EAdd (EAdd (EMul (EInt 1, EInt (-8)), EMul (EInt 2, EInt 1))),

EMul (EInt (-7), EInt 3))),

EMul (EAdd (EAdd (EInt 6, EInt 9), EAdd (EInt 1, EInt (-5))),

EInt (-6))),

EMul

(EMul

(EAdd (EInt (-10),

EAdd (EMul (EInt (-1), EInt 3), EAdd (EInt 1, EInt (-1))))),

EAdd (EAdd (EMul (EInt 1, EInt (-8)), EMul (EInt 2, EInt 1))),

EMul (EInt (-7), EInt 3))),

EMul (EInt 5, EAdd (EMul (EMul (EInt 2, EInt 0), EInt (-7)), EInt (-5))))))

Correct value

1 pt

(EMul

(EAdd

(EMul (EInt (-10),

EAdd (EMul (EInt (-1), EInt 3), EAdd (EInt 1, EInt (-1))))),

EAdd (EAdd (EMul (EInt 1, EInt (-8)), EMul (EInt 2, EInt 1))),

Computing factorize (EMul (EInt 8, EMul (EAdd (EInt 3, EInt 5), EMul (EInt 6, EMul (EInt 2, EInt (-7))))))	1 pt
Correct value (EMul (EInt 8, EMul (EAdd (EInt 3, EInt 5), EMul (EInt 6, EMul (EInt 2, EInt (-7))))))	
Computing factorize (EMul (EInt 4, EInt (-8)))	
Correct value (EMul (EInt 4, EInt (-8)))	1 pt
Computing factorize (EAdd (EMul (EMul (EAdd (EAdd (EAdd (EInt 7, EInt (-2)), EInt (-3)), EMul (EAdd (EInt 4, EInt 7), EInt (-8))), EMul (EMul (EAdd (EInt (-6), EInt (-1)), EAdd (EInt 5, EInt (-9))), EMul (EAdd (EInt 3, EInt (-9)), EInt 0))), EMul (EMul (EMul (EInt (-4), EInt 3), EAdd (EInt (-8), EAdd (EInt (-8), EInt 6))), EInt 0)), EMul (EMul (EAdd (EAdd (EAdd (EInt 7, EInt (-2)), EInt (-3)), EMul (EAdd (EInt 4, EInt 7), EInt (-8))), EMul (EMul (EAdd (EInt (-6), EInt (-1)), EAdd (EInt 5, EInt (-9))), EMul (EAdd (EInt 3, EInt (-9)), EInt 0))), EMul (EMul (EMul (EInt 6, EInt (-2)), EAdd (EMul (EInt (-7), EInt 3), EMul (EInt 4, EInt (-6))))), EAdd (EMul (EMul (EInt (-3), EInt (-7)), EInt (-10)), EAdd (EInt 2, EInt 5))))))	
Correct value (EMul (EMul (EAdd (EAdd (EAdd (EInt 7, EInt (-2)), EInt (-3)), EMul (EAdd (EInt 4, EInt 7), EInt (-8))), EMul (EMul (EAdd (EInt (-6), EInt (-1)), EAdd (EInt 5, EInt (-9))), EMul (EAdd (EInt 3, EInt (-9)), EInt 0))), EAdd (EMul (EMul (EMul (EInt (-4), EInt 3), EAdd (EInt (-8), EAdd (EInt (-8), EInt 6))), EInt 0)), EMul (EMul (EMul (EInt 6, EInt (-2)), EAdd (EMul (EInt (-7), EInt 3), EMul (EInt 4, EInt (-6))))), EAdd (EMul (EMul (EInt (-3), EInt (-7)), EInt (-10)), EAdd (EInt 2, EInt 5))))))	1 pt
Computing factorize (EMul (EAdd (EMul (EInt (-6), EInt 6), EMul (EMul (EInt (-3), EInt 5), EInt (-2))), EMul (EInt 1, EInt (-2))))	
Correct value (EMul (EAdd (EMul (EInt (-6), EInt 6), EMul (EMul (EInt (-3), EInt 5), EInt (-2))), EMul (EInt 1, EInt (-2))))	1 pt
Computing factorize (EAdd (EMul (EMul (EMul (EMul (EInt 2, EAdd (EInt (-5), EInt (-4))), EMul (EInt 9, EMul (EInt 3, EInt 8))), EInt (-4)), EAdd (EInt (-10), EMul (EMul (EInt (-1), EInt (-3)), EAdd (EInt (-8), EInt 8)))), EMul (EMul (EMul (EMul (EInt 2, EAdd (EInt (-5), EInt (-4))), EMul (EInt 9, EMul (EInt 3, EInt 8))), EInt (-4)), EAdd (EMul (EMul (EMul (EInt (-7), EInt 7), EInt 7), EAdd (EMul (EInt 8, EInt (-6)), EInt (-6))), EMul (EAdd (EAdd (EInt 2, EInt 8), EInt (-6)), EMul (EMul (EInt 6, EInt (-9)), EAdd (EInt 2, EInt (-8))))))	
Correct value (EMul (EMul (EMul (EMul (EInt 2, EAdd (EInt (-5), EInt (-4))),	1 pt

```

    EMul (EMul (EInt (-1), EInt (-3)), EAdd (EInt (-8), EInt 8))),
  EAdd
    (EMul (EMul (EMul (EInt (-7), EInt 7), EInt 7),
      EAdd (EMul (EInt 8, EInt (-6)), EInt (-6))),
    EMul (EAdd (EAdd (EInt 2, EInt 8), EInt (-6)),
      EMul (EMul (EInt 6, EInt (-9)), EAdd (EInt 2, EInt (-8))))))
Computing
factorize
  (EAdd
    (EMul (EAdd (EInt (-7), EInt (-10)),
      EMul (EInt (-3),
        EMul (EAdd (EInt 2, EAdd (EInt 7, EInt 6)), EInt (-3)))),
    EMul (EAdd (EInt (-7), EInt (-10)), EInt (-6)))
Correct value 1 pt
  (EMul (EAdd (EInt (-7), EInt (-10)),
    EAdd
      (EMul (EInt (-3), EMul (EAdd (EInt 2, EAdd (EInt 7, EInt 6)), EInt (-3))),
      EInt (-6)))
Computing
factorize
  (EMul (EMul (EInt (-10), EMul (EInt (-4), EInt 4)),
    EAdd (EInt (-5), EAdd (EMul (EInt (-6), EInt (-6)), EInt (-2))))
Correct value 1 pt
  (EMul (EMul (EInt (-10), EMul (EInt (-4), EInt 4)),
    EAdd (EInt (-5), EAdd (EMul (EInt (-6), EInt (-6)), EInt (-2))))
v Exercise 4: expand Completed, 10 pts
Found expand with compatible type.
Computing
expand
  (EMul
    (EAdd (EInt 3,
      EMul (EInt (-1),
        EAdd (EMul (EInt (-3), EInt (-4)), EMul (EInt 8, EInt (-5))))),
    EAdd
      (EMul (EInt (-1),
        EMul (EAdd (EInt 7, EMul (EInt (-9), EInt 7)), EInt 7)),
      EMul (EMul (EInt (-6), EInt (-5)),
        EAdd (EMul (EInt 3, EInt 4), EAdd (EAdd (EInt (-6), EInt 4), EInt 8))))))
Correct value 1 pt
  (EAdd
    (EMul
      (EAdd (EInt 3,
        EMul (EInt (-1),
          EAdd (EMul (EInt (-3), EInt (-4)), EMul (EInt 8, EInt (-5))))),
      EMul (EInt (-1), EMul (EAdd (EInt 7, EMul (EInt (-9), EInt 7)), EInt 7))),
    EMul
      (EAdd (EInt 3,
        EMul (EInt (-1),
          EAdd (EMul (EInt (-3), EInt (-4)), EMul (EInt 8, EInt (-5))))),
      EMul (EMul (EInt (-6), EInt (-5)),
        EAdd (EMul (EInt 3, EInt 4), EAdd (EAdd (EInt (-6), EInt 4), EInt 8))))))
Computing
expand
  (EAdd (EInt (-6),
    EAdd (EMul (EMul (EInt 3, EInt 2), EAdd (EInt (-8), EInt 1)),
      EMul (EInt 9, EMul (EInt (-5), EInt (-2))))))
Correct value 1 pt
  (EAdd (EInt (-6),
    EAdd (EMul (EMul (EInt 3, EInt 2), EAdd (EInt (-8), EInt 1)),
      EMul (EInt 9, EMul (EInt (-5), EInt (-2))))))
Computing
expand
  (EMul
    (EMul (EInt (-7),
      EMul (EAdd (EAdd (EInt 8, EInt 1), EInt 3),
        EMul (EInt (-3), EMul (EInt (-9), EInt 4)))),
    EAdd
      (EMul
        (EAdd (EAdd (EMul (EInt (-3), EInt (-4)), EMul (EInt (-5), EInt 1)),
          EAdd (EMul (EInt 1, EInt 8), EAdd (EInt (-10), EInt 3))),
        EInt 9),
      EMul
        (EAdd (EAdd (EMul (EInt (-7), EInt (-4)), EMul (EInt 3, EInt (-6))),
          EMul (EInt 3, EInt 3)),
        EAdd (EMul (EMul (EInt 0, EInt 2), EMul (EInt (-1), EInt 2)), EInt 4))))
Correct value 1 pt
  (EAdd
    (EMul
      (EMul (EInt (-7),
        EMul (EAdd (EAdd (EInt 8, EInt 1), EInt 3),
          EMul (EInt (-3), EMul (EInt (-9), EInt 4)))),
      EMul
        (EAdd (EAdd (EMul (EInt (-3), EInt (-4)), EMul (EInt (-5), EInt 1)),
          EAdd (EMul (EInt 1, EInt 8), EAdd (EInt (-10), EInt 3))),
        EInt 9),
    EMul
      (EAdd (EAdd (EMul (EInt (-3), EInt (-4)), EMul (EInt (-5), EInt 1)),
        EAdd (EMul (EInt 1, EInt 8), EAdd (EInt (-10), EInt 3))),
      EInt 9)),

```

```

EMul
(EAdd (EAdd (EMul (EInt (-7), EInt (-4)), EMul (EInt 3, EInt (-6))),
EMul (EInt 3, EInt 3))),
EAdd (EMul (EMul (EInt 0, EInt 2), EMul (EInt (-1), EInt 2)), EInt 4))))

```

Computing

expand

```

(EMul (EAdd (EInt 5, EInt (-10)),
EAdd
(EAdd (EMul (EInt (-10), EMul (EInt 6, EInt (-6))),
EMul (EAdd (EAdd (EInt (-3), EInt 7), EInt (-2)), EInt 0)),
EAdd (EInt 7,
EAdd (EMul (EAdd (EInt 5, EInt 6), EAdd (EInt (-9), EInt 4)),
EAdd (EAdd (EInt 8, EInt 8), EAdd (EInt (-6), EInt (-6)))))))

```

Correct value

1 pt

```

(EAdd
(EMul (EAdd (EInt 5, EInt (-10)),
EAdd (EMul (EInt (-10), EMul (EInt 6, EInt (-6))),
EMul (EAdd (EAdd (EInt (-3), EInt 7), EInt (-2)), EInt 0))),
EMul (EAdd (EInt 5, EInt (-10)),
EAdd (EInt 7,
EAdd (EMul (EAdd (EInt 5, EInt 6), EAdd (EInt (-9), EInt 4)),
EAdd (EAdd (EInt 8, EInt 8), EAdd (EInt (-6), EInt (-6)))))))

```

Computing

expand

```

(EMul
(EMul
(EMul (EMul (EAdd (EInt 4, EInt 8), EInt (-5)),
EAdd (EInt (-2), EInt (-8))),
EAdd (EMul (EInt 6, EMul (EInt 6, EInt (-3))),
EAdd (EInt (-6), EAdd (EInt 4, EInt 2)))),
EAdd
(EAdd
(EAdd (EAdd (EAdd (EInt (-7), EInt (-6)), EInt (-2)),
EMul (EAdd (EInt (-6), EInt 9), EMul (EInt 5, EInt (-8)))),
EAdd (EMul (EInt (-8), EMul (EInt 0, EInt (-8))),
EMul (EAdd (EInt (-8), EInt (-1)), EInt (-6))),
EAdd
(EAdd (EAdd (EInt (-4), EMul (EInt 6, EInt (-5))),
EAdd (EMul (EInt 3, EInt 9), EInt 7)),
EMul (EMul (EAdd (EInt (-8), EInt 4), EInt (-5)),
EMul (EInt 4, EInt 5))))))

```

Correct value

1 pt

```

(EAdd
(EMul
(EMul
(EMul (EMul (EAdd (EInt 4, EInt 8), EInt (-5)),
EAdd (EInt (-2), EInt (-8))),
EAdd (EMul (EInt 6, EMul (EInt 6, EInt (-3))),
EAdd (EInt (-6), EAdd (EInt 4, EInt 2)))),
EAdd
(EAdd (EAdd (EAdd (EInt (-7), EInt (-6)), EInt (-2)),
EMul (EAdd (EInt (-6), EInt 9), EMul (EInt 5, EInt (-8)))),
EAdd (EMul (EInt (-8), EMul (EInt 0, EInt (-8))),
EMul (EAdd (EInt (-8), EInt (-1)), EInt (-6))))),
EMul
(EMul
(EMul (EMul (EAdd (EInt 4, EInt 8), EInt (-5)),
EAdd (EInt (-2), EInt (-8))),
EAdd (EMul (EInt 6, EMul (EInt 6, EInt (-3))),
EAdd (EInt (-6), EAdd (EInt 4, EInt 2)))),
EAdd
(EAdd (EAdd (EInt (-4), EMul (EInt 6, EInt (-5))),
EAdd (EMul (EInt 3, EInt 9), EInt 7)),
EMul (EMul (EAdd (EInt (-8), EInt 4), EInt (-5)), EMul (EInt 4, EInt 5))))))

```

Computing

expand

```

(EMul
(EMul (EAdd (EInt 6, EMul (EInt (-7), EInt (-3))),
EMul (EInt (-10), EMul (EInt 3, EInt 5))),
EInt (-10)))

```

Correct value

1 pt

```

(EMul
(EMul (EAdd (EInt 6, EMul (EInt (-7), EInt (-3))),
EMul (EInt (-10), EMul (EInt 3, EInt 5))),
EInt (-10)))

```

Computing

expand

```

(EAdd (EAdd (EMul (EInt (-9), EAdd (EInt 0, EInt 9)), EInt 1),
EMul (EInt (-4), EMul (EInt 2, EMul (EInt (-1), EInt (-9))))))

```

Correct value

1 pt

```

(EAdd (EAdd (EMul (EInt (-9), EAdd (EInt 0, EInt 9)), EInt 1),
EMul (EInt (-4), EMul (EInt 2, EMul (EInt (-1), EInt (-9))))))

```

Computing

expand

```

(EMul
(EAdd

```

```

(EAdd (EAdd (EInt 6, EAdd (EInt (-3), EInt (-7))),
  EMul (EMul (EMul (EInt (-8), EInt 6), EInt (-9)),
    EMul (EMul (EInt (-8), EInt 9), EInt (-4)))),
  EAdd (EInt (-10), EInt 9)))

```

Correct value

1 pt

```

(EAdd
  (EMul
    (EAdd
      (EMul (EMul (EInt 1, EInt (-8)),
        EAdd (EMul (EInt 2, EInt (-5)), EAdd (EInt 7, EInt (-6)))),
      EMul (EMul (EAdd (EInt 9, EInt (-10)), EInt 5), EAdd (EInt 8, EInt 8))),
    EAdd (EAdd (EInt 6, EAdd (EInt (-3), EInt (-7))),
      EMul (EMul (EMul (EInt (-8), EInt 6), EInt (-9)),
        EMul (EMul (EInt (-8), EInt 9), EInt (-4))))),
    EMul
      (EAdd
        (EMul (EMul (EInt 1, EInt (-8)),
          EAdd (EMul (EInt 2, EInt (-5)), EAdd (EInt 7, EInt (-6)))),
          EMul (EMul (EAdd (EInt 9, EInt (-10)), EInt 5), EAdd (EInt 8, EInt 8))),
        EAdd (EInt (-10), EInt 9)))

```

Computing

expand

```

(EMul
  (EMul (EInt 1, EMul (EAdd (EInt 9, EMul (EInt (-1), EInt 2)), EInt (-9))),
  EAdd
    (EMul (EInt 4,
      EMul (EMul (EMul (EInt (-1), EInt 1), EInt 8),
        EAdd (EMul (EInt 1, EInt (-8)), EInt 8))),
    EAdd
      (EMul (EMul (EAdd (EInt 0, EInt (-3)), EMul (EInt 9, EInt 5)), EInt 9),
        EAdd (EInt 0, EInt 5))))

```

Correct value

1 pt

```

(EAdd
  (EMul
    (EMul (EInt 1, EMul (EAdd (EInt 9, EMul (EInt (-1), EInt 2)), EInt (-9))),
    EMul (EInt 4,
      EMul (EMul (EMul (EInt (-1), EInt 1), EInt 8),
        EAdd (EMul (EInt 1, EInt (-8)), EInt 8))),
    EMul
      (EMul (EInt 1, EMul (EAdd (EInt 9, EMul (EInt (-1), EInt 2)), EInt (-9))),
      EAdd
        (EMul (EMul (EAdd (EInt 0, EInt (-3)), EMul (EInt 9, EInt 5)), EInt 9),
          EAdd (EInt 0, EInt 5))))

```

Computing

expand

```

(EMul
  (EAdd
    (EMul (EInt (-8), EMul (EAdd (EInt 4, EInt 2), EMul (EInt 1, EInt 2))),
    EMul (EMul (EMul (EInt 6, EInt 5), EAdd (EInt (-9), EInt (-9))),
      EMul (EInt 6, EInt (-2)))),
    EAdd
      (EAdd
        (EAdd (EMul (EMul (EInt 6, EInt 8), EInt 4),
          EMul (EAdd (EInt 6, EInt (-4)), EMul (EInt 6, EInt (-2)))),
          EAdd (EInt 3, EAdd (EMul (EInt 9, EInt (-9)), EInt 9))),
        EAdd (EAdd (EInt (-7), EMul (EInt 0, EAdd (EInt (-1), EInt 0))),
          EMul (EInt 9, EMul (EMul (EInt 0, EInt 5), EMul (EInt 4, EInt 8))))))

```

Correct value

1 pt

```

(EAdd
  (EMul
    (EAdd
      (EMul (EInt (-8), EMul (EAdd (EInt 4, EInt 2), EMul (EInt 1, EInt 2))),
      EMul (EMul (EMul (EInt 6, EInt 5), EAdd (EInt (-9), EInt (-9))),
        EMul (EInt 6, EInt (-2)))),
    EAdd
      (EAdd (EMul (EMul (EInt 6, EInt 8), EInt 4),
        EMul (EAdd (EInt 6, EInt (-4)), EMul (EInt 6, EInt (-2)))),
        EAdd (EInt 3, EAdd (EMul (EInt 9, EInt (-9)), EInt 9))),
    EMul
      (EAdd
        (EMul (EInt (-8), EMul (EAdd (EInt 4, EInt 2), EMul (EInt 1, EInt 2))),
        EMul (EMul (EMul (EInt 6, EInt 5), EAdd (EInt (-9), EInt (-9))),
          EMul (EInt 6, EInt (-2)))),
        EAdd (EAdd (EInt (-7), EMul (EInt 0, EAdd (EInt (-1), EInt 0))),
          EMul (EInt 9, EMul (EMul (EInt 0, EInt 5), EMul (EInt 4, EInt 8))))))

```

v Exercise 5: simplify

Completed, 10 pts

Found simplify with compatible type.

Computing simplify (EMul (EInt (-8), EInt (-9)))

Correct value (EMul (EInt (-8), EInt (-9)))

1 pt

Computing

simplify

(EMul (EMul (EMul (EInt (-8), EMul (EInt 0, EInt (-9))), EInt 6), EInt 0))

Correct value (EInt 0)

1 pt

<pre>      EAdd (EMul (EInt (-10), EInt (-8)), EMul (EInt 4, EInt (-6))),       EAdd (EInt (-8), EInt 8)))</pre>	1 pt
<pre>Correct value (EMul   (EAdd (EInt 3,     EAdd (EMul (EInt (-10), EInt (-8)), EMul (EInt 4, EInt (-6))),     EAdd (EInt (-8), EInt 8)))</pre>	1 pt
<pre>Computing simplify (EMul (EInt 3, EInt 1)) Correct value (EInt 3)</pre>	1 pt
<pre>Computing simplify   (EMul (EInt 1,     EMul (EInt (-4),       EAdd (EAdd (EInt 2, EInt (-1)), EInt 3),       EAdd (EMul (EInt (-10), EInt (-3)), EAdd (EInt 0, EInt (-3))))))</pre>	1 pt
<pre>Correct value   (EMul (EInt (-4),     EMul (EAdd (EAdd (EInt 2, EInt (-1)), EInt 3),       EAdd (EMul (EInt (-10), EInt (-3)), EAdd (EInt 0, EInt (-3))))))</pre>	1 pt
<pre>Computing simplify   (EAdd     (EAdd (EInt 0,       EAdd (EAdd (EMul (EInt 2, EInt (-2)), EInt (-10)), EInt (-3)),       EInt 0))</pre>	1 pt
<pre>Correct value (EAdd (EInt 0, EAdd (EAdd (EMul (EInt 2, EInt (-2)), EInt (-10)), EInt (-3))))</pre>	1 pt
<pre>Computing simplify   (EMul (EInt 0,     EMul (EInt (-9),       EMul (EMul (EInt 6, EInt (-10)),         EMul (EInt 0, EAdd (EInt (-10), EInt (-10))))))</pre>	1 pt
<pre>Correct value (EInt 0)</pre>	1 pt
<pre>Computing simplify   (EAdd (EInt 0,     EAdd (EAdd (EInt 5, EAdd (EInt (-2), EAdd (EInt 8, EInt (-10)))),       EMul (EMul (EInt (-10), EInt (-7)), EAdd (EInt (-3), EInt 5))))</pre>	1 pt
<pre>Correct value (EAdd (EAdd (EInt 5, EAdd (EInt (-2), EAdd (EInt 8, EInt (-10)))),   EMul (EMul (EInt (-10), EInt (-7)), EAdd (EInt (-3), EInt 5))))</pre>	1 pt
<pre>Computing simplify   (EMul (EInt 0,     EMul       (EAdd (EAdd (EInt (-6), EAdd (EInt 3, EInt (-7))),         EAdd (EMul (EInt (-7), EInt (-7)), EMul (EInt (-6), EInt 0))),       EAdd (EAdd (EInt (-5), EInt (-2)),         EMul (EMul (EInt 1, EInt 2), EInt (-2))))))</pre>	1 pt
<pre>Correct value (EInt 0)</pre>	1 pt
<pre>Computing simplify   (EMul (EInt 1,     EAdd (EMul (EInt (-10), EAdd (EInt (-7), EAdd (EInt (-2), EInt 0))),       EInt 9)))</pre>	1 pt
<pre>Correct value (EAdd (EMul (EInt (-10), EAdd (EInt (-7), EAdd (EInt (-2), EInt 0))), EInt 9))</pre>	1 pt





Rechercher un cours

