Algebraic Simplifier

Write an F# function `simplify` (in *hw5.fsx*) that simplifies an algebraic expression. The expression is using a similar expression class used for symbolic differentiation in class. It removes the divide and power operators but adds a second variable Y. It is shown below (and is in *hw5_prep.fsx*).

```
type Expression =
    | X
    | Y
    | Const of float
    | Neg of Expression
    | Add of Expression * Expression
    | Sub of Expression * Expression
    | Mul of Expression * Expression
```

To start, download the ***hw5_prep.fsx*** (https://seattleu.instructure.com/courses/1602042/files/67684843/download?download_frd=1) , rename it to *hw5.fsx*, and use it as a starting point for your assignment. Edit the initial header comment to have your own name and remarks and then search for all the *FIXME* strings to see where you need to add materials. Your file *hw5.fsx* now contains an `Expression` type, an `exprToString` function that converts an expression to a more readable string format, and contains tests for the examples shown in this document.

Your `simplify` function must perform the following algebraic simplifications:

- Addition involving two numbers. (See t1 in the provided file for a test of this simple example.)
  - `Add (Const 5.0, Const 3.0)` -> `Const 8.0`
- Subtraction involving two numbers. (t2)
  - `Sub (Const 5.0, Const 3.0)` -> `Const 2.0`
- Multiplication involving two numbers. (t3)
  - `Mul (Const 5.0, Const 3.0)` -> `Const 15.0`
- Negation involving a number. (t4 and t5)
  - `Neg (Const 4.0)` -> `Const -4.0`
  - `Neg (Const -9.0)` -> `Const 9.0`
- Addition with zero. (t6 and t7)
  - `Add (X, Const 0.0)` -> `X`
  - `Add (Const 0.0, Y)` -> `Y`
- Subtraction with zero. (t8 and t9)
  - `Sub (X, Const 0.0)` -> `X`
  - `Sub (Const 0.0, Y)` -> `Neg Y`
- Subtraction with identical terms. (t10)
  - `Sub (Y, Y)` -> `Const 0.0`
- Multiplication with zero. (t11 and t12)
  - `Mul (X, Const 0.0)` -> `Const 0.0`

    - `Mul (Const 0.0, Y)` -> `Const 0.0`
- Multiplication with one. (t13 and t14)
    - `Mul (X, Const 1.0)` -> `X`
    - `Mul (Const 1.0, Y)` -> `Y`
- Double negation. (t15)
    - `Neg (Neg X)` -> `X`

Like symbolic differentiation, `simplify` must be applied recursively. For example, simplifying the following expression requires three simplifications to arrive at 0.

```
Sub (Mul (Const 1.0, X), Add (X, Const 0.0))
```

```
Sub (X, Add (X, Const 0.0)) // simplify the multiply
Sub (X, X) // simplify the add
Const 0.0 // simplify the subtract
```

This expression is test t16 in the provided file.

Additional examples from the interactive interpreter:

```
> simplify (Add (Mul (Const 4.0, Const 3.0), Sub (Const 11.0, Const 5.0)));;
val it : Expression = Const 18.0
```

```
> simplify (Sub (Sub (Add (X, Const 1.0), Add (X, Const 1.0)), Add (Y, X)));;
val it : Expression = Neg (Add (Y,X))

> simplify (Sub (Const 0.0, Neg (Mul (Const 1.0, X))));;
val it : Expression = X

> simplify (Mul (Add (X, Const 1.0), Neg (Sub (Mul (Const 2.0, Y), X))));;
val it : Expression = Mul (Add (X,Const 1.0), Neg (Sub (Mul (Const 2.0, Y), X)))
```

The same four examples in algebraic form:

Original Expression	***Simplified Expression***
(4 × 3) + (11 − 5)	18
$(x + 1) - (x + 1) - (y + x)$	$-(y + x)$

$0 - -(1 \times x)$	x
$(x + 1) \times -(2 \times y) - x$	$(x + 1) \times -(2 \times y) - x$ *[no simplification possible]*

These four examples are tests t17-t20 in the provided file.

Additional notes:

- For comparisons of the same expression, only look for exact matches. You do not have to look for algebraically equivalent expressions. For instance Add (Const 3.0, X) is an exact match of Add (Const 3.0, X) but not Add (X, Const 3.0).
- Some input expressions may not be able to be simplified (such as the last expression in the table above) – simply return the non-simplified expression.
- Do not modify the `Expression` type and `exprToString`. Any modifications will cause tests to fail, possibly resulting in a very low score.
- Do not consider testing to be complete with the 20 provided tests. Be sure to add your own tests.

HW5 Rubric

Criteria	Ratings			Pts
20 provided tests 2 point for each test	**40 pts** **All tests pass**	**40 to >0.0 pts** **Some tests pass and some tests fail**	**0 pts** **All tests fail**	40 pts
10 additional instructor-provided tests 1 points for each test	**10 pts** **All tests pass**	**10 to >0.0 pts** **Some tests pass and some tests fail**	**0 pts** **All tests fail**	10 pts
Additional Deductions	**0 pts** **Additional deductions** represented using negative points		**0 pts** **No additional deductions**	0 pts
Total Points: 50				