

School of Computer Science and Software Engineering

CSE3322 Programming Languages and Implementation

Assignment 2

Due 12 noon Friday 12th of September

The purpose of this assignment is to teach more advanced ML programming and an appreciation of functional programming.

A polynomial has form

$$2.0x^2y^3x^1 + 3.0 + (-2.0)y^3x^3 + 1.0x + 5.0$$

A polynomial is composed of *terms*, e.g. $2.0x^2y^3x^1$, 3.0 , $-2.0y^3x^3$, $1.0x$ and 5.0 . Terms are composed of a *coefficient* and *factors*. E.g. $-2.0y^3x^3$ has coefficient -2.0 and factors y^3 and x^3 . Note that x and y are variables: In general variables can be arbitrary character strings.

Given a *valuation*, i.e. a mapping from variables to their values, we can evaluate a polynomial. For example, if we evaluate the above polynomial with the valuation $[x \mapsto 2.0, y \mapsto 1.0]$ we obtain 10.0 .

We may also simplify polynomials by simplifying terms in the polynomial and collecting terms with the same factors together. The above polynomial can be simplified to $1.0x + 8.0$.

You are to write a **Polynomial** structure in Standard ML for manipulating polynomials. You should call the file containing the structure `poly.ml`. It should provide the following:

- Type definitions for `factor`, `term`, and `polynomial` which respectively represent a factor, term, and polynomial. [1 mark]
- Functions for constructing a factor, a term and a polynomial:

```
factor: string * int -> factor
term: real * factor list -> term
poly: term list -> polynomial
```

The function `factor` takes a variable name and an integer power and returns the corresponding factor, the function `term` takes a real coefficient and a list of factors and returns the corresponding term, and the function `poly` takes a list of terms and returns the corresponding polynomial. These should not simplify the factors, terms or polynomials. For instance,

```
val term1 = term(2.0, [factor("x",2),factor("y",3),factor("x",1)]);
val term2 = term(3.0, []);
val term3 = term(~2.0, [factor("y",3),factor("x",3)]);
val term4 = term(1.0, [factor("x",1)]);
val term5 = term(5.0, []);
val p = poly([term1,term2,term3,term4,term5]);
```

binds `p` to the representation for the polynomial given in the example above. [1 mark]

- A function

```
toString: polynomial -> string
```

for returning a string describing a polynomial. For instance `toString p` should return the string

```
"2.0 x^2 y^3 x^1 + 3.0 + ~2.0 y^3 x^3 + 1.0 x^1 + 5.0"
```

[1 mark]

- A function

```
add: polynomial*polynomial -> polynomial
```

where `add(p1,p2)` is the polynomial obtained by adding the terms in `p1` and `p2`. It does not need to perform simplification. For instance,

```
add(poly([term1,term2,term3]),poly([term4,term5]))
```

will return the example polynomial given above. [1 mark]

- A function

```
scale: real -> polynomial -> polynomial
```

where `scale r p` is the polynomial obtained by multiplying the coefficient of all terms in `p` by `r`. [1 mark]

- A function

```
eval: (string -> real) -> polynomial -> real
```

where `eval v p` is the result of evaluating `p` with `v`. For instance,

```
eval (fn v => if v = "x" then 2.0 else if v = "y" then 1.0 else 0.0) p
```

should return 10.0. You might want to make use of the library function `Real.Math.pow`. [2 marks]

- A function which takes a polynomial and returns a simplified version of the polynomial:

```
simplify: polynomial -> polynomial
```

The function should

- simplify terms by merging multiple variable occurrences in the same term into a single occurrence,
- simplify terms by removing factors with a power of 0,
- merge terms with the same factors into a single term and
- remove terms with 0.0 coefficients.

For instance `simplify p` should return the representation for $1.0x + 8.0$. Implementing `simplify` is more difficult—leave it till last. You may wish to sort the factors and terms to aid in simplification and you may wish to define `simplify` in terms of a function to simplify terms and a function which then simplifies a polynomial given simplified terms. [2 marks]

- Define a **Polynomial** structure which groups the above functions and type definitions. It should hide the definitions of the types and export only the functions detailed above (although you will need to define some hidden functions.) [1 mark]

Submission Instructions

The above exercises contribute 10% to your total CSE3322 mark.

Submission will be electronic. You should use `/cs/cc/bin/submit` to submit the file **poly.ml**. The assignment name is **ass2**.

The assignment is due **12 noon Friday 12th September 2003**.

Your programs will be marked on correctness, style, efficiency and clarity.

Assignments handed in after the due date will attract a late penalty of 5% per day unless special consideration applies or there has been prior agreement in writing from the lecturer. No submission will be accepted after 12 noon Friday 19th of September.

Your submission must include the following declaration at the top of **poly.ml**. If it does not your assignment will not be marked and you will receive 0 for the assignment.

(* MONASH UNIVERSITY, School of Computer Science and Software Engineering
Student Declaration for CSE3322 Submission

I #YOUR NAME#, ID: #YOUR ID NUMBER# declare that this submission is
my own work and has not been copied from any other source without attribution.
I acknowledge that severe penalties exist for any copying of code without
attribution, including a mark of 0 for this assessment.

*)

Unless otherwise advised your assignment marks for CSE3322 are to be listed on the 3rd year notice board next to your student ID. Of course your name will not be on the list, only your ID and assignment mark. If you do not wish your mark and student ID to appear on this list please contact your lecturer for the subject by Tuesday 2nd September to make alternative arrangements.