Helena Josol <hmj27@bath.ac.uk>
Advanced Programming Principles

Lisp implementation choice
I used Euscheme.

Testing
Euscheme lacks a unit testing framework so I implemented a simple one as an exercise and for me to use. I wrote a unit test for each function that contributes to the overall functionality (i.e. the polynomial operations and auxiliary/helper functions). A unit test consists of several assert statements. The syntax of an assert statement is

```
(assert-equal <expected-result> <actual-result>)
; argument order does not really matter but this is
; the convention I followed in my test cases
```

How to run the tests
```
$ ./test
```

How to use
The syntax to make a variable is

```
(make-var <symbol> <power>)
; (make-var ‘x 1)
```

The syntax to make a term is

```
(make-term <coefficient> ‘(<make-var-statements>))
; (make-term 2 ‘((make-var ‘x 2) (make-var ‘y 1))) => 2x^2y
```

The syntax to make a polynomial is

```
(make-poly ‘(<make-term-statements>))
; (make-poly
;   ‘((make-term 1 ‘((make-var ‘x 2)))
;     (make-term 2 ‘((make-var ‘x 1) (make-var ‘y 1)))
;     (make-term 1 ‘((make-var ‘y 2)))))
; => x^2 + 2xy + y^2
```

Examples are given in the test cases, and I apologise for the clunkiness.

Data structures and representations
A variable is represented as a cons pair

```
(make-var ‘x 1) ; => ‘(x . 1)
```

A term is represented as a list consisting of a number (coefficient) and a list of cons pairs (variables)

```
(make-term 2 ‘((make-var ‘x 2) (make-var ‘y 1)))
; => ‘(2 ((x . 2) (y . 1)))
```

A polynomial is represented as a list of lists (terms)

```
(make-poly
  ‘((make-term 1 ‘((make-var ‘x 2)))
    (make-term 2 ‘((make-var ‘x 1) (make-var ‘y 1)))
    (make-term 1 ‘((make-var ‘y 2)))))
; => ‘( (1 ((x . 2)))  (2 ((x . 1) (y . 1)))  (1 ((y . 2))) )
```

Method

- A polynomial is constructed by creating each term in turn.
- Each term is constructed by creating its variable list and prepending the coefficient.
- A variable list is simplified and sorted lexicographically (e.g. $w^1z^3y^2x^4x^{-2}y^{-1}z^{-2}w^{-1} = x^2yz$).
- A polynomial is simplified and sorted lexicographically by the first variable’s symbol of each term and by its power in descending order (this is only for convenience when testing).

Notes

- The main file is `polynomial-arithmetic.lisp`. The other files are loaded by it.
- For convenience, the function `polypretty` pretty prints polynomials `(polypretty <polynomial>) ; (polypretty (make-poly ‘((make-term …))))`