Comp 333 Project #3 (20 pts) Revised: March 10 See yellow highlights.

Due: Tuesday March 17

GENERAL DIRECTIONS: In this project you will create a set of Scheme definitions to implement a polynomial system. You must use Dr Racket. Your source code must be named polynomials.rkt . Neatness counts and so does indented code that is easy to read with helpful variable names.

[VERY IMPORTANT NOTE] You may only use the Scheme functions or expressions that we have discussed in class or that appear in my power point slides. I want you solve the problems "from scratch" by working only with the most basic Scheme functions and expressions and recursion. Do not use vectors or the do form. Use define only to define functions at the global level and to define lists for testing your code. **See last page for list of functions and forms you may use.**

PROJECT: Write a set of Scheme definitions that implement a polynomial system as defined below.

1. A *(value exponent)* tuple is called a **term**. The value may be any real number but the exponent must be a nonnegative integer. A **polynomial** will be represented as a list terms. For example
   $4x^2$ + -5 x + 10 can be represented by the list '((4 2) (-5 1) (10 0)) . The list may not be sorted on the exponent so that $4x^2$ + -5 x+ 10 can also be represented by the list '((10 0) (-5 1) (4 2)) . We will always use the symbol x as the polynomial unknown.

   Any list of (value exponent) pairs represents a polynomial. The polynomial might not be simplified. For example, the list '((6 4) (10 0) (8 2) (-3 5) (3 2) (3 0)) represents the polynomial $6x^4$ + 10 + $8x^2$ + $-3x^5$ + $3x^2$ + 3

2. Write the following Scheme functions. The functions should all be in the same file. Use recursion for items d,e, i, m . You may use recursion for the others, if needed.
   a. (coeff t) // returns the value of the term t
   b. (expon t) //returns the exponent of the term t
   c. (printTerm t) //prints the term t in the format *value x ^ exponent*.
      Examples: (4 2) will be printed as 4x^2
      (7 0) will be printed as 7
      (-4 1) will be printed as -4x
   d. (printpoly p) // prints the polynomial p using printTerm with a **+** between the terms.
      For example, '((2.13 3) (1.5 4) (6 3) (4 1) (-3 0)) should be printed as
      2.13x^3 + 1.5x^4 + 6x^3 + 4x + -3
   e. (evalpoly p v) // returns the value of polynomial when x = v.
   f. (GT t1 t2) //returns true if exponent of t1 > exponent of t2, else false
   g. (EQExp? t1 t2) //returns true if t1 and t2 have the same exponent, else false
   h. Show that (sort p GT) returns polynomial p sorted on exponents.

i. (simplify p) // returns a polynomial equivalent to p which is simplified, in that all terms have different exponents and are listed in decreasing exponent order. For example, ‘((6 4) (8 2) (-3 5) (3 2) (13 0)) should return ‘((-3 5) (6 4) (11 2) (13 0)) .
[Hint: Sort p and then call a recursive function that simplifies a sorted polynomial]

j. (addpoly p1 p2) // returns a simplified polynomial which is the sum of p1 and p2
[Hint: first append the two polynomials]

k. (subtractpoly p1 p2) //returns a simplified polynomial which is the difference between p1 and p2 (i.e., p1 – p2). [Hint: p1 – p2 = p1 + (-1)p2]

l. (multiplyterms t1 t1) returns the product of the terms t1 and t2. For example, (multiplyterms ‘(3 5) ‘(-2 3)) returns ‘(-6 8)

m. (multiplytermpoly t1 p1) // returns the simplified polynomial that is the product of t1 and p1

n. (multiplypoly p1 p2) //returns the simplified polynomial that is the product of p1 and p2. This is a challenge.

3. You may create other helper Scheme functions.

4. Test your code thoroughly. Run Instructor test cases. I will post them on March 10.

5. For functions that are incomplete or not working , you must have a” placeholder function ” in your .rkt file that returns “incomplete or not working function” whenever it is called. For example,

```
(define  multiplypoly
  (lambda (p1 p2)
     “incomplete or missing function”  ) )
```

Turn in:

a. Hard copy: Submit (1) polynomials.rkt file; (2) runs of posted instructor test cases. Label everything. Names must be listed in the polynomial.rkt file. Use a Cover page as in Project 2Electronic Copy: Upload a single source file called polynomials.rkt with your racket definitions to Moodle. Your source file should contain as a comment your name, date and Project. Submit only one file person. (Due at 7am on March 17)

List of Scheme functions and forms you can use for this project:

- define, lambda, if, cond, cons,car, cdr, list , member, list-ref
- predicates : null? list? equal? string? number? member? and others we have used in class
- arithmetic operators , relational operators, logical operators used in class ,
- sort, map, filter, length, reverse, append, last , let, let*, letrec, print, begin, newline, display, set!
- If there is a function you think should be added to this list, you need to get your instructor's approval.
- Added functions: expt, apply , min, max, unless