Comp 333 Project #2 (30 pts) Fall 2016

Due: Nov 15

**GENERAL DIRECTIONS:** This is an individual project. 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. Your function names and arguments must match the specifications below. Otherwise my test cases will fail.

**[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, recursion, map, filter, foldr and reduce (from class). Do not use vectors or set!. 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 **polynomial** will be represented as a non-empty list of coefficients, starting with the highest degree term with a non-zero coefficient unless the only term is 0. Missing coefficients are denoted with a 0, as they should be.

For example

5 is represented by ‘(5)

0 is represented by ‘(0)

4x2 + -5 x + 10.2 is represented by the list ‘( 4 -5 10.2 ) .

5x7 + -2.3x3 + 10x + 9 is represented as ‘( 5 0 0 0 -2.3 0 10 9)

We will always use the symbol x as the polynomial unknown.

1. Write the following Scheme functions. The functions should all be in the same .rkt file.
   1. (degree p) //returns the degree of the polynomial

(degree ‘( 7 9 4 2 1 2 ) ) returns 5

* 1. (displayTerm coeff exponent) //display one term as it would be printed in displayPoly

//Use the built-in procedure display

(displayTerm 3 5) should print 3x^5

(displayTerm 1 1) should print x

(displayTerm 1 7) should print x^7

(displayTerm 2.3 0) should print 2.3

* 1. (displayPoly p ) // displays the polynomial p in fairly natural fashion. For example,

‘( 2 0 1 -3.6 - 5 ) should be displayed as 2x^4 + x^2 + -3.6x + - 5

‘( 1 1) should be displayed as x + 1

Put a + between all of the terms. Leave two spaces between terms and + sign. Do not print the terms with a 0 coefficient. Print the x^1 term with simply a coefficient followed by x. Use recursion.

* 1. (polynomial? z) tests if z is a polynomial. It must be a non-empty list of numbers starting with a non- zero number unless the list is ‘( 0 )

(polynomial? ‘( 4 6 2 4 7) ) returns #t

(polynomial? ‘ ( (4 5 ) ) ) returns #f

(polynomial? ‘( 0 5 6 7) ) returns #f

(polynomial? ‘(0) ) returns #t

(polynomial? ‘( 4 6 “art” ) ) returns #f

(polynomial? ‘()) returns #f

* 1. (evalPoly p v ) // returns the value of polynomial when x = v.

Use recursion or use map and foldl ( or reduce from class).

* 1. (multiplyPolyByConstant p a ) // returns the polynomial a\*p . Use map

(multiplyPolyByConstant ‘( 1 2 3 ) 5 ) returns ‘(5 10 15)

Multiplying by 0 should return ‘( 0 )

* 1. (multiplyPolyByX p ) //adds a 0 to end of the list of coefficients

(multiplyPolyByX ‘( 3 4 5) ) returns ‘( 3 4 5 0)

* 1. (addPoly p1 p2) // returns a polynomial which is the sum of p1 and p2

For example (addPoly ‘( 3 -4 5 1) ‘( 5 6 2) ) returns ‘( 3 1 11 3 )

Use map.

[Hint: If the polynomials p1 and p2 are not the same degree, extend the smaller one with zeros. Then use map on 2 lists. Make sure that there are no leading zeros in the polynomial returned unless the result is ‘(0).]

* 1. (subtractPoly p1 p2) //returns a polynomial which is the difference of p1 and p2. Use map. Make sure there are no leading zeros in the polynomial returned unless the result is ‘(0).
  2. (multiplyPoly p1 p2) //returns the polynomial that is the product of p1 and p2. This is a challenge. Break it up into smaller functions. Make sure there are no leading zeros in the polynomial returned.

1. You may create other helper Scheme functions.
2. Test your code thoroughly. Run Instructor test cases. I will post them on Nov 8.
3. 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:**

* 1. Hard copy: Submit (1) polynomials.rkt file; (2) Code and output of posted instructor test cases. Label everything. Your source file should contain your name, due date and Project # as a comment. Use a Cover page as in Project 1. Due in class on Nov 15
  2. Electronic Copy: Upload a single source file called polynomials.rkt with your racket definitions to Moodle. Your source file should contain your name, date and Project# as a comment. Submit only one file .rkt file with all of your code, including helper functions. (Due at 8am on Nov 15)

**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, foldr, foldl, length, reverse, append, last , let, let\*, letrec, print, begin, newline, display, , expt, string-append, reduce(from class), range

If there is a function you think should be added to this list, you need to get your instructor’s approval.