CPSC 3740—Fall 2023
Course Project

Due: December 8, 2023 11:55pm

1 Description

In this group project, you will implement a set of functions that will perform operations on polynomials in Racket. You can assume that the given input is syntactically correct.

2 Polynomial Representation

There are two representations of polynomials that you must support. For illustration, consider the polynomial

$$p(x) = 3x^5 + 2x^3 + 5x^2 + 1.$$

Dense representation: the polynomial is a list of coefficients of x^0, x^1, x^2, etc. The polynomial $p(x)$ above is represented as the list

```
(1 0 5 2 0 3).
```

Trailing 0 terms are not allowed unless $p(x) = 0$. In that case, the representation of $p(x)$ is

```
(0)
```

Sparse representation: the polynomial is a list of pairs of coefficients and power of x. The polynomial $p(x)$ above is represented as the list

```
((1 0) (5 2) (2 3) (3 5))
```

The entries are sorted in increasing power of x. No entry in this list should have 0 as its first element, unless $p(x) = 0$. The sparse representation of the zero polynomial is `((0 0))`.

You may assume that all coefficients are integers.

3 Required Functions

The following functions must be implemented, and they must work regardless of which representation is given. If all input polynomials have the same representation, the output polynomial must be in the same representation. Otherwise, the output should be in dense representation.

You may choose to implement additional helper functions.

✓ **is-dense?:** is the input a polynomial in dense representation?

✓ **is-sparse?:** is the input a polynomial in sparse representation?

✓ **to-dense:** given a polynomial, converts to dense representation if necessary.

✓ **to-sparse:** given a polynomial, converts to sparse representation if necessary.

✓ **degree:** returns the degree of the polynomial (degree of the zero polynomial is negative infinity: `-inf.0`).

✓ **is-zero?:** is the input the zero polynomial?

✓ **coeff:** takes a polynomial and a power k, and returns the coefficient of the polynomial corresponding to x^k.

✓ **eval:** takes a polynomial $p(x)$ and a value k, returns the result of $p(k)$.

add: adds two polynomials and returns the sum.

subtract: subtracts one polynomial from another and returns the difference.

multiply: multiplies one polynomial by another and returns the product.

quotient: divide a polynomial $p(x)$ by $q(x)$ and returns the quotient. You may assume $q(x) \neq 0$.

remainder: divide a polynomial $p(x)$ by $q(x)$ and returns the remainder. You may assume $q(x) \neq 0$.

derivative: returns the derivative of the polynomial.

gcd: given two polynomials, returns their greatest common divisor. If the greatest common divisor is non-zero, ensures that the leading coefficient (coefficient corresponding to highest power of x) is 1 by multiplying by an appropriate constant.

To normalize the leading coefficient of a polynomial, you need to divide all the coefficients of the polynomial by the leading coefficient. The leading coefficient is the coefficient of the term with the highest degree of the polynomial, that is, the coefficient of the term with the highest power of x. After dividing all the coefficients by the leading coefficient, the leading coefficient will become 1. This normalization is useful because it makes the greatest common divisor unique up to multiplication by a constant

4 Written Report

You will also submit a written report (at most 5 pages) addressing the following elements:

- How did you deal with different representations for each function?
- How did you implement each function
- Are there any limitations/bugs?
- How did you test your functions? Include sample test runs illustrating each of the required functions.

5 Grading

Your project will be graded on functionality and readability. It is expected that you make use of functions to organize your code, and that you document each function and its parameters. You are also expected to document the main idea of any non-trivial algorithm or data structures involved.

Functionality: 65%

> **is-sparse/dense:** 4%
>
> **to-sparse/dense:** 10%
>
> **degree:** 3%
>
> **is-zero:** 3%
>
> **coeff:** 3%
>
> **eval:** 4%
>
> **add/subtract:** 8%
>
> **multiply/quotient/remainder:** 15%
>
> **derivative:** 5
>
> **gcd:** 10%

Documentation: 15%

Report: 20%

6 Submission

Your project should be submitted on Moodle (instructions will be posted). Your implementation should be stored in a file named `polynomial.rkt`.

7 Group Membership

You may choose to complete this project in groups of up to 3 people. **Please indicate to the instructor by November 1, 2023 the group composition.** All members of the group will receive the same grade. It is your responsibility to ensure each member contribute to the project. Note that the grading scheme is the same regardless of the size of the group.

8 Hints

- This are many aspects in this project. You may want to start by implementing one functionality at a time and ensuring that the functionality is correct before continuing.
- It is recommended that you start with the simpler functionalities, and build the more complicated ones by using the simpler ones.
- However, before you begin coding you need to think about how you would deal with polynomials in both representations in each of the required functions.
- Efficiency is not a big concern in this assignment. You should not expect input polynomials with degrees higher than 20.