Institute for Advancing Intelligence, TCG CREST
(TCG Centres for Research and Education in Science and Technology)

Introduction to Programming and Data Structures
Ph.D. Coursework: First year, First Semester (Session: 2024-25)
Assignment #05

Full Marks: 200 | Instructor: Dr. Laltu Sardar
Clarification Deadline: **2024-Oct-31** | Submission Deadline: **2024-Nov-03**

Instructions

1. Errors must be handled in all possible functions used, whether from libraries or written by yourself
2. Function names and variable names should clearly describe their purpose.
3. Write the program in such a way, that program does not fails.
4. Magic numbers (like 100 in `array[100]`) should not be hard-coded across the programs. Instead define them as macros (E.g. `#define ARRAY_SIZE 100` and later `array[ARRAY_SIZE]`).

Assignment Overview

In this assignment, you will implement a set of operations on polynomials, such as addition, subtraction, multiplication, and evaluation. You will represent a polynomial as a linked list where each node contains the coefficient and the exponent of a term. You will then write functions to perform various polynomial operations.

Problem Statement

1. Polynomial Representation

- Represent a polynomial as a linked list of nodes. Each node should contain two pieces of data: the coefficient of the term and the exponent.
- Example: For the polynomial $5x^3 + 4x^2 - 2x + 7$, the linked list will have four nodes, where each node stores the coefficient and exponent.

2. Required Operations

- **Polynomial Creation:**
  - Write a function to create a polynomial by reading the coefficients and exponents of terms from the user. The input should be of the form:

```
Enter the number of terms: 3
Enter coefficient and exponent: 5 3
Enter coefficient and exponent: 4 2
Enter coefficient and exponent: -2 1
```

- **Polynomial Display:**

  - Write a function to display a polynomial in a readable form, such as $5x^3 + 4x^2 - 2x + 7$.

- **Polynomial Addition:**
  - Write a function to add two polynomials and return the resulting polynomial. The addition should combine terms with the same exponents.
- **Polynomial Subtraction:**
  - Write a function to subtract one polynomial from another and return the resulting polynomial. As with addition, terms with the same exponents should be combined.
- **Polynomial Multiplication:**
  - Write a function to multiply two polynomials and return the resulting polynomial. In multiplication, the exponents of terms are added, and the coefficients are multiplied.
- **Polynomial Evaluation:**
  - Write a function to evaluate a polynomial for a given value of x. This function should take a polynomial and a value of x, and compute the result using the polynomial's terms.
- **Polynomial Division:** Write a function to implement polynomial division, returning both the quotient and the remainder. You can use long division for this.

Specifications

- Use **structures** to represent the terms of a polynomial.
- Use **dynamic memory allocation** to create and manipulate polynomials of varying sizes.
- **Input/Output**: Your program should be able to accept multiple polynomials and perform operations on them.
- Ensure **memory safety** by freeing dynamically allocated memory after use.

Function Prototypes

```
typedef struct Term {
    int coefficient;
    int exponent;
    struct Term* next;
} Term;

Term* create_polynomial();
void display_polynomial(Term* poly);
Term* add_polynomials(Term* poly1, Term* poly2);
Term* subtract_polynomials(Term* poly1, Term* poly2);
Term* multiply_polynomials(Term* poly1, Term* poly2);
int evaluate_polynomial(Term* poly, int x);
void free_polynomial(Term* poly);
void divide_polynomials(Term* poly1, Term* poly2, Term** quotient, Term** remainder);
```

Detailed Requirements

1. Polynomial Creation

- Dynamically allocate memory for each term.
- Insert the terms in the linked list in decreasing order of exponent. This will simplify the polynomial operations.

2. Polynomial Addition/Subtraction

- Traverse both polynomial linked lists and add/subtract terms with the same exponent.
- The result should also be a linked list of terms sorted by decreasing exponents.

3. Polynomial Multiplication

- For each term in the first polynomial, multiply it with every term in the second polynomial and store the result in a new polynomial.
- Combine terms with the same exponent.

4. Polynomial Evaluation

Use the formula result $= \sum (\text{coefficient} \cdot x^{\text{exponent}})$ for all terms to compute the result for a given value of x.

Deliverables

1. A well-commented C source code implementing the above operations.
2. A test suite that demonstrates the functionality of each of the polynomial operations with at least two different sets of polynomials.
3. A document (PDF or Word) explaining the approach, challenges faced, and a brief explanation of your solution for each part of the assignment.

Sample Input and Output

Input:

```
Enter the first polynomial:
Enter the number of terms: 3
Enter coefficient and exponent: 3 2
Enter coefficient and exponent: 5 1
Enter coefficient and exponent: 6 0

Enter the second polynomial:
Enter the number of terms: 3
Enter coefficient and exponent: 4 3
Enter coefficient and exponent: 2 2
Enter coefficient and exponent: -3 1
```

Output:

```
First Polynomial: 3x^2 + 5x + 6
Second Polynomial: 4x^3 + 2x^2 - 3x

Addition Result: 4x^3 + 5x^2 + 2x + 6
Subtraction Result: -4x^3 + x^2 + 8x + 6
Multiplication Result: 12x^5 + 26x^4 + 11x^3 - 9x^2 - 15x
Evaluation of First Polynomial at x = 2: 28
```

[200]