# Lab 01

# Data Structures

# BS DS Fall 2024 Morning/Afternoon

## Objective

• Reinforcing the concepts of pointers and Problem-Solving Guidelines and Focusing on Exception Handling Mechanism.
• For a refresher of OOP, Array of Objects, Dynamic Memory Allocation, Shallow/Deep Copy phenomena.

## Task 1: Polynomial ADT in C++

A polynomial is an equation of the form: **$P(x) = a_0x^{e0} + a_1x^{e1} + \dots + a_nx^{en}$** where $a_i$ belongs to Coefficient $e_i$ belongs to Exponent. Your target is to store this form of equation in Computer and be able to perform the operations listed below.

Selection of data members for this ADT is upto you but you should be able to justify the need of all data members.

*You should always implement the Def-Ctor/Copy-Ctor/Dtor/= methods in case of dynamic memory allocation to class data members.*

While coding keep in mind the concept behind Problem Solving Guidelines and basic principles that we studied in OOP and also take care of Coding Convention. You must also use exception handling mechanism wherever needed.

### Public Functions

**1. Ctor**
Constructor method to initialize the Polynomial object.

**2. addTerm(coefficient, power)**
Add a new term xpower term in the polynomial and set its coefficient.

**3. getDegree()**
Returns the degree of a polynomial. For example, when p1 = 4x5 + 7x3 - x2 + 9, p1 degree is 5.

**4. getCoefficient(power)**
Returns the coefficient of the x power term. For example, when p1 = $4x^5 + 7x^3 - x^2 + 9$ p1.getCoefficient (3) is 7

**5. operator(value)**

For example, given a polynomial in "p1", p1(x) will evaluate the polynomial for the given value of x.

**6. Dtor / Copy-Ctor / =**

**7. Arithmetic Operation +**

Add two polynomials. For instance, assuming p1 = $4x^5 + 7x^3 - x^2 + 9$ and p2 = $6x^4 + 3x^2 + 2x$, then the answer of p1 + p2 must be $4x^5 + 6x^4 + 7x^3 + 2x^2 + 2x + 9$.

**8. derivative()**

Return a polynomial that is the derivative of the given polynomial. For instance, assuming p1 = $4x^5 + 7x^3 - x^2 + 9$, the derivative polynomial would be $20x^4 + 21x^2 - 2x$.

**9. antiDerivative()**

Return a polynomial that is the anti-derivative of the given polynomial. For instance, assuming p1 = $20x^4 + 21x^2 - 2x$, the derivative polynomial would be $4x^5 + 7x^3 - x^2 + C$, Where C is a random constant value.

**10. addToCoefficient(coefficient, power)**

Adds the given amount to the coefficient of the xpower term. For example, if p1 = $20x^4 + 21x^2 - 2x$, then p1.addToCoefficient (2.5, 3) changes the coefficient of $x^3$ to 9.5

**11. clear()**

Set the coefficient of all terms in the polynomial to zero.

**12. setCoefficient(newCoefficient, power)**

Sets the coefficient of the xpower term with newCoeffcient. For instance, p1.setCoefficient (-3, 7) produces the polynomial p1 = $-3x^7 + 4x^5 + 7x^3 - x^2 + 9$. Note that if the term does not exist in the polynomial, it is added.

**13. Arithmetic Operations *, -**

Perform multiplication and subtraction of two polynomials.

**14. ostream& operator<<(ostream &out, const Polynomial &p)**

Used with Polynomial objects to display their string representations. For example $20x^4 + 21x^2 - 2x$ will be displayed in the following format **20x^4 + 21x^2 +  -2^x**