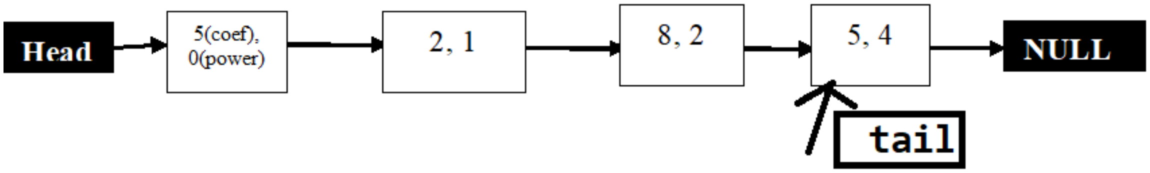
An-Najah National University
Department of Computer Engineering
Data Structure 10636211
Hw # 2

Deadline: 8/4/2021

Q1: Polynomials can be represented as a sorted linked list. Consider the Polynomial $F(x) = 5 + 2\ x + 9\ x^2 + 5\ x^3$.

The Polynomial can be represented as a list of nodes which contain the coefficient and the power of x. The list is sorted in increasing order of power of x.

Write a complete C++ code that do the following:

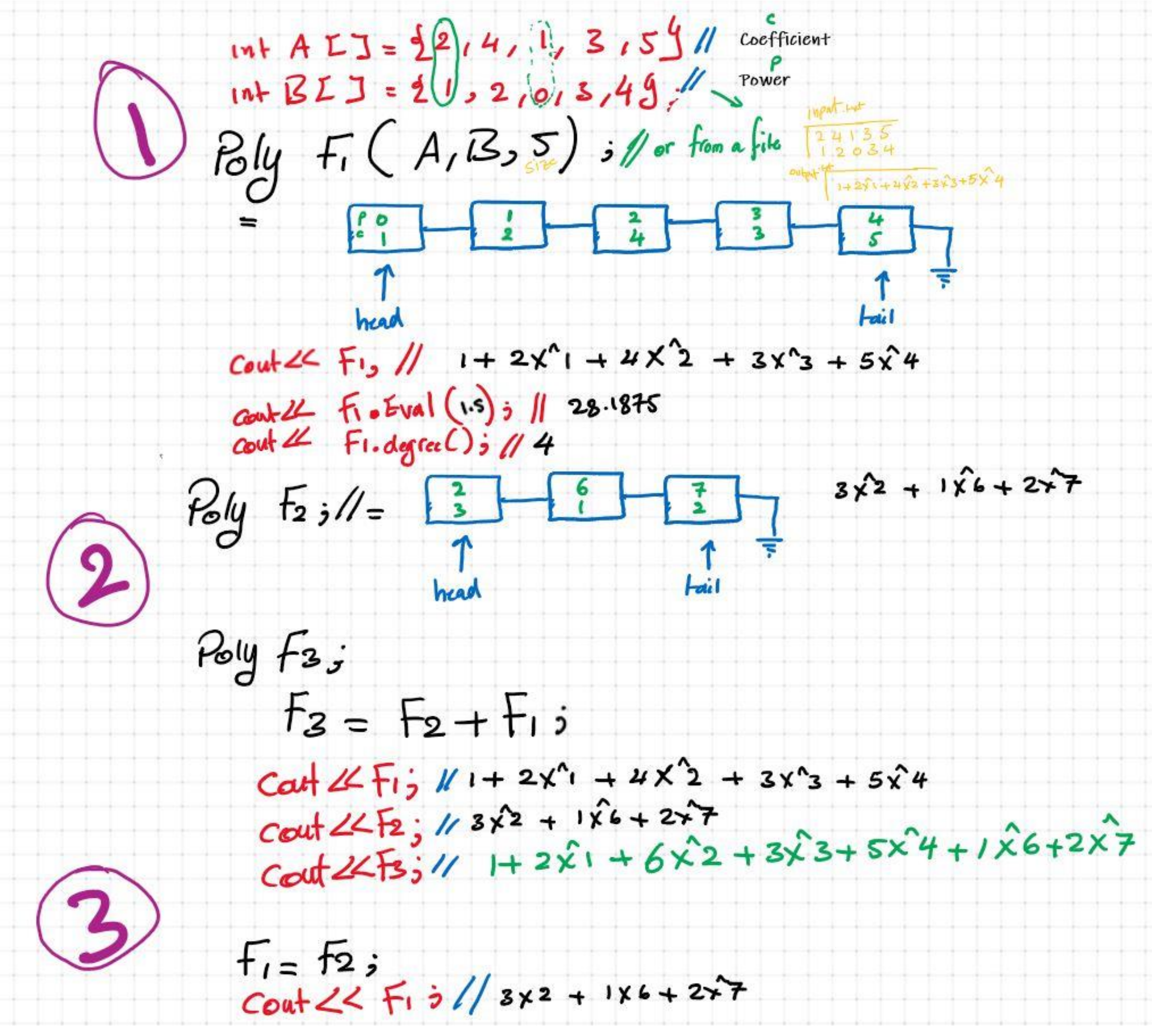


1- Declare the structure **NODE** which represent coefficient and power as integers (private).

2- The class **Poly** that represents a single linked list with **head** and **tail** pointers (private). The class should include the following functions:

- Default **Constructor**.
- **Constructor** that takes integer arrays for coefficients and power. (Coeff_arr, Pow_arr)
- **Destructor**
- Write function to insert a node to the linked list sorted based on power. Make sure if there is a term with **the same entered power, then you should add them**.
- Write function to delete a node from the linked list **based on the power.**
- Write the function to overload the operator >> such that it reads and creates all nodes that represent the polynomial from a file or a user Keyboard.
- Write the function to overload the operator << to print the polynomial represented by the list in a nice format on the console and output file.
- Write the function **Eval** which returns the result of evaluation the polynomial for a value as an input parameter.
- Add two **Poly** objects using **+ operator**.
- Write function that takes a linked list and then return the degree of polynomial (degree = highest power).
- Merge two **Poly** objects into one object. Passed as a parameter. Check examples below.

Result example:

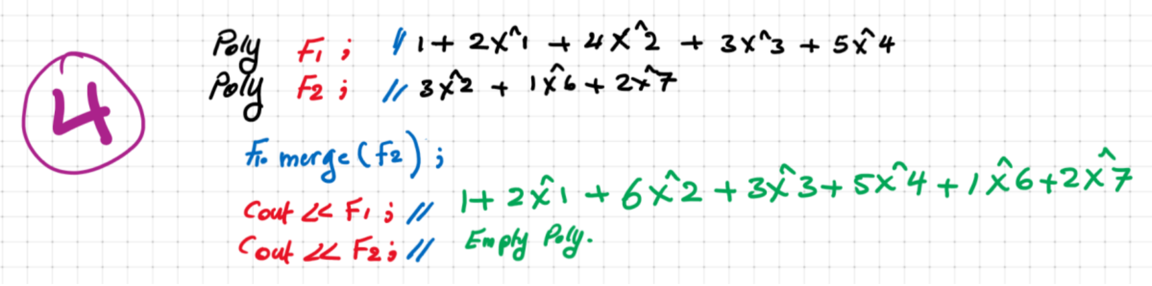


```
int A[] = {2, 4, 1, 3, 5} // Coefficient
int B[] = {1, 2, 0, 3, 4}; // Power
Poly F1(A, B, 5); // or from a file
=
cout << F1, // 1+ 2x^1 + 4x^2 + 3x^3 + 5x^4
cout << F1.Eval(1.5); // 28.1875
cout << F1.degree(); // 4

Poly F2; //=                3x^2 + 1x^6 + 2x^7

Poly F3;
    F3 = F2 + F1;
    cout << F1; // 1+ 2x^1 + 4x^2 + 3x^3 + 5x^4
    cout << F2; // 3x^2 + 1x^6 + 2x^7
    cout << F3; // 1+ 2x^1 + 6x^2 + 3x^3 + 5x^4 + 1x^6 + 2x^7

F1 = F2;
cout << F1; // 3x2 + 1x6 + 2x^7
```



```
Poly F1; // 1+ 2x^1 + 4x^2 + 3x^3 + 5x^4
Poly F2; // 3x^2 + 1x^6 + 2x^7
  F1.merge(F2);
  cout << F1; // 1+ 2x^1 + 6x^2 + 3x^3 + 5x^4 + 1x^6 + 2x^7
  cout << F2; // Empty Poly.
```