Practical No 3

Aim : Write A Program To Implement Polynomials Using Linked List.

- Represent polynomials using linked lists.
- Perform polynomial addition and subtraction by merging lists.
- Use structured representation to reinforce node manipulation.

```
class Node:
    def __init__(self,c,v,e,nextNode=None):
        self.c = c
        self.v = v
        self.e = e
        self.nextNode = nextNode

class Polynomial:
    def __init__(self):
        self.head = None

    def print(self):
        curr = self.head
        equation = ""
        while curr:
            if curr.c == 0:
                equation += ""
            elif curr.c > 0:
                equation += (f" + {curr.c}{curr.v}^{curr.e} ")
            else:
                equation += (f" {curr.c}{curr.v}^{curr.e} ")
            curr = curr.nextNode
        print(equation)

    def insert(self,c,v,e):
        # If Coeficient Is Zero
        if c == 0:
            return True

        newNode = Node(c,v,e)

        # Case 1 : Linked List Is Empty
        if self.head is None:
            self.head = newNode
            return True

        # Case 2 : Insert At The Beginning
        if e > self.head.e:
            newNode.nextNode = self.head
```

```
            self.head = newNode
            return True
        # If Exponential Already Exist
        elif e == self.head.e and v == self.head.v:
            self.head.c += c
            if self.head.c == 0:
                self.head = self.head.nextNode
            return True

        # Case 3 : Insert At The Middle Or In The End
        prev = self.head
        curr = self.head.nextNode

        while curr:
            # Finding Position
            if e > curr.e:
                break
            # If Exponential Already Exist
            elif e == curr.e and v == curr.v:
                curr.c += c
                if curr.c == 0:
                    curr = curr.nextNode
                return True
            # Shifting Pointer
            prev = curr
            curr = curr.nextNode

        # Insert Node
        newNode.nextNode = curr
        prev.nextNode = newNode
        return True

    def add(self,e):
        curr = e.head
        while curr:
            self.insert(curr.c,curr.v,curr.e)
            curr = curr.nextNode
        return True

    def subtract(self,e):
        curr = e.head
        while curr:
            self.insert(-curr.c,curr.v,curr.e)
            curr = curr.nextNode
        return True
```

```
# First Equation
p = Polynomial()
p.insert(2,"x",3)
p.insert(2,"x",5)
p.insert(2,"x",4)
print("First Equation : ",end="");p.print()

# Second Equation
e = Polynomial()
e.insert(2,"x",1)
e.insert(2,"x",2)
e.insert(-3,"x",5)
print("Second Equation : ",end="");e.print()

# Adding Equation
print("")
print("Before Adding : ",end="");p.print()
p.add(e)
print("After Adding : ",end="");p.print()

# Subtracting Equation
print("")
print("Before Subtracting : ",end="");p.print()
p.subtract(e)
print("After Subtracting : ",end="");p.print()
```

```
>>>
============ RESTART: C:\MyWork\CS-Sem-3\DataStructure\polynomial.py ===========
First Equation :  + 2x^5  + 2x^4  + 2x^3
Second Equation :  -3x^5  + 2x^2  + 2x^1

Before Adding :  + 2x^5  + 2x^4  + 2x^3
After Adding :  -1x^5  + 2x^4  + 2x^3  + 2x^2  + 2x^1

Before Subtracting :  -1x^5  + 2x^4  + 2x^3  + 2x^2  + 2x^1
After Subtracting :  + 2x^5  + 2x^4  + 2x^3
```