Lab 1

DSA BS DS Fall 2022

Task 1

A polynomial is an equation of the form: $P(x) = a_0x^{e0} + a_1x^{e1} + \ldots + a_nx^{en}$ 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 need to implement the following methods of your Polynomial ADT:

1. `__init__(self)`
Constructor method to initialize the Polynomial object.

2. `addTerm(self, coefficient, power)`
Add a new term x power term in the polynomial and sets its coefficient.

3. `getDegree(self)`
Returns the degree of a polynomial. For example, when p1 = $4x^5 + 7x^3 - x^2 + 9$, p1 degree is 5.

4. `getCoefficient(self, 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. `evaluate(self, value)`
Evaluate the polynomial for a given value of the variable. For example when p1 = $4x^5 + 7x^3 - x^2 + 9$, p.evaluate(2) is 285.

6. `__add__(self, other)`
Perform addition of two polynomials.

7. `derivative(self)`
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 $20x^4 + 21x^2 - 2x$.

8. `antiDerivative(self, constant)`
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 constant.

9. **addToCoefficient(self, coefficient, power)**
Adds the given amount to the coefficient of the xpower term. For example, when p1 = $4x^5 + 7x^3 - x^2 + 9$ p1.addToCoefficient (2, 3) changes the coefficient of x^3 to 9.

10. `clear(self)`

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

11. `setCoefficient(self, 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.

12. `__mul__(self, other)`
Perform multiplication of two polynomials.

13. `__sub__(self, other)`
Perform subtraction of two polynomials.
14. `__str__(self)`
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**

Use the following driver code to test your ADT

```
p1 = Polynomial()
p1.addTerm(4, 5)
p1.addTerm(7, 3)
p1.addTerm(-1, 2)
p1.addTerm(9, 0)

p2 = Polynomial()
p2.addTerm(6, 4)
p2.addTerm(3, 2)
p2.addTerm(2, 1)

print ("P1: ", p1)
print ("P2: ", p2)
result = p1 + p2
print("Addition result:", result)

result_derivative = p1.derivative()
print("Derivative result:", result_derivative)

result_anti_derivative = result_derivative.antiDerivative(3)
print("Anti-derivative result:", result_anti_derivative)
```

The output of the following code should be like this:

```
P1:  4x^5 + 7x^3 + -1x^2 + 9
P2:  6x^4 + 3x^2 + 2x
Addition result: 4x^5 + 7x^3 + 2x^2 + 9 + 6x^4 + 2x
Derivative result: 20x^4 + 21x^2 + -2x
Anti-derivative result: 4.0x^5 + 7.0x^3 + -1.0x^2 + 3
```

Task 2

The greatest common divisor (GCD) of two positive integers A and B is the largest positive integer that divides both A and B without leaving a remainder. It is also called the highest common factor (HCF).

For example, the greatest common factor of 15 and 10 is 5, since both the numbers can be divided by 5. i.e. 15/5 = 3. 10/5 = 2.

Write a recursive function to find the GCD of two numbers.

```
def calculateGCD(A, B):
    # The definition of your recursive function




# Example inputs
examples = [[12, 18], [25, 15], [40, 60]]
for e in examples:
    A, B = e
    gcd = calculateGCD(A, B)
    print("GCD of", A, "and", B, "is:", gcd)
```

The output of the following code should be like this:

```
GCD of 15 and 5 is: 5
GCD of 18 and 6 is: 6
GCD of 60 and 40 is: 20
```