class Node {

int coefficient;

int exponent;

Node next;

Node(int coefficient, int exponent) {

this.coefficient = coefficient;

this.exponent = exponent;

this.next = null;

}

}

public class Polynomial {

Node head;

int degree;

int numTerms;

public void insertTerm(int coefficient, int exponent) {

if (coefficient == 0) {

return; // Don't add terms with a coefficient of 0

}

Node newTerm = new Node(coefficient, exponent);

if (head == null || exponent > head.exponent) {

newTerm.next = head;

head = newTerm;

} else {

Node current = head;

Node previous = null;

while (current != null && exponent < current.exponent) {

previous = current;

current = current.next;

}

if (current != null && exponent == current.exponent) {

current.coefficient += coefficient;

if (current.coefficient == 0) {

if (previous != null) {

previous.next = current.next;

} else {

head = current.next;

}

}

} else {

newTerm.next = current;

if (previous != null) {

previous.next = newTerm;

} else {

head = newTerm;

}

}

}

numTerms++;

if (exponent > degree) {

degree = exponent;

}

}

public Polynomial add(Polynomial otherPoly) {

Polynomial result = new Polynomial();

Node current1 = this.head;

Node current2 = otherPoly.head;

while (current1 != null || current2 != null) {

if (current1 != null && (current2 == null || current1.exponent > current2.exponent)) {

result.insertTerm(current1.coefficient, current1.exponent);

current1 = current1.next;

} else if (current2 != null && (current1 == null || current2.exponent > current1.exponent)) {

result.insertTerm(current2.coefficient, current2.exponent);

current2 = current2.next;

} else {

int newCoefficient = current1.coefficient + current2.coefficient;

result.insertTerm(newCoefficient, current1.exponent);

current1 = current1.next;

current2 = current2.next;

}

}

return result;

}

public Polynomial subtract(Polynomial otherPoly) {

Polynomial result = new Polynomial();

Node current1 = this.head;

Node current2 = otherPoly.head;

while (current1 != null || current2 != null) {

if (current1 != null && (current2 == null || current1.exponent > current2.exponent)) {

result.insertTerm(current1.coefficient, current1.exponent);

current1 = current1.next;

} else if (current2 != null && (current1 == null || current2.exponent > current1.exponent)) {

result.insertTerm(-current2.coefficient, current2.exponent);

current2 = current2.next;

} else {

int newCoefficient = current1.coefficient - current2.coefficient;

result.insertTerm(newCoefficient, current1.exponent);

current1 = current1.next;

current2 = current2.next;

}

}

return result;

}

public Polynomial multiply(Polynomial otherPoly) {

Polynomial result = new Polynomial();

if (head == null || otherPoly.head == null) {

return result; // If either polynomial is empty, the result is also an empty polynomial

}

Node current1 = this.head;

while (current1 != null) {

Node current2 = otherPoly.head;

while (current2 != null) {

int newCoefficient = current1.coefficient \* current2.coefficient;

int newExponent = current1.exponent + current2.exponent;

result.insertTerm(newCoefficient, newExponent);

current2 = current2.next;

}

current1 = current1.next;

}

return result;

}

public void display() {

Node current = head;

while (current != null) {

System.out.print(current.coefficient + "x^" + current.exponent);

current = current.next;

if (current != null) {

System.out.print(" + ");

}

}

System.out.println();

}

public static void main(String[] args) {

Polynomial poly1 = new Polynomial();

poly1.insertTerm(3, 2);

poly1.insertTerm(5, 1);

poly1.insertTerm(4, 0);

Polynomial poly2 = new Polynomial();

poly2.insertTerm(2, 3);

poly2.insertTerm(1, 1);

poly2.insertTerm(7, 0);

System.out.println("Polynomial 1:");

poly1.display();

System.out.println("Polynomial 2:");

poly2.display();

Polynomial sum = poly1.add(poly2);

System.out.println("Sum:");

sum.display();

Polynomial difference = poly1.subtract(poly2);

System.out.println("Difference:");

difference.display();

Polynomial product = poly1.multiply(poly2);

System.out.println("Product:");

product.display();

}

}