NAME - AVANISH RAJ SRIVASTAVA
ROLL - BT22CSH031

```
#include <iostream>
#include <cmath>
using namespace std;

class Term {
public:
    int coef;
    int exp;
    Term* next;

    Term(int coefficient, int exponent) : coef(coefficient), exp(exponent), next(nullptr) {}
};

class Polynomial {
private:
    Term* head;

public:
    Polynomial() {
        head = new Term(0, -1);
        head->next = head;
    }

    void readPolynomial() {
        int numTerms;
        cout << "Enter the number of terms: ";
        cin >> numTerms;

        for (int i = 0; i < numTerms; i++) {
            int coefficient, exponent;
            cout << "Enter coefficient and exponent for term " << i + 1 << ": ";
            cin >> coefficient >> exponent;

            insertTerm(coefficient, exponent);
        }
    }

    void insertTerm(int coefficient, int exponent) {
        Term* newNode = new Term(coefficient, exponent);
        Term* current = head;

        while (current->next != head && current->next->exp >= exponent) {
            current = current->next;
        }
```

```
    newNode->next = current->next;
    current->next = newNode;
}

void displayPolynomial() {
    Term* current = head->next;
    bool isFirstTerm = true;

    while (current != head) {
        if (current->coef != 0) {
            if (!isFirstTerm && current->coef > 0) {
                cout << "+";
            }

            if (current->exp == 0) {
                cout << current->coef;
            } else {
                cout << current->coef << "x^" << current->exp;
            }

            isFirstTerm = false;
        }
        current = current->next;
    }

    cout << endl;
}

void addPolynomials(Polynomial& a, Polynomial& b) {
    Term* termA = a.head->next;
    Term* termB = b.head->next;
    Polynomial result;

    while (termA != a.head && termB != b.head) {
        if (termA->exp > termB->exp) {
            result.insertTerm(termA->coef, termA->exp);
            termA = termA->next;
        } else if (termA->exp < termB->exp) {
            result.insertTerm(termB->coef, termB->exp);
            termB = termB->next;
        } else {
            int sum = termA->coef + termB->coef;
            if (sum != 0) {
                result.insertTerm(sum, termA->exp);
            }
            termA = termA->next;
            termB = termB->next;
```

```
        }
    }

    while (termA != a.head) {
        result.insertTerm(termA->coef, termA->exp);
        termA = termA->next;
    }

    while (termB != b.head) {
        result.insertTerm(termB->coef, termB->exp);
        termB = termB->next;
    }

    *this = result;
}

void subtractPolynomials(Polynomial& a, Polynomial& b) {
    Polynomial negB;
    Term* current = b.head->next;

    while (current != b.head) {
        negB.insertTerm(-current->coef, current->exp);
        current = current->next;
    }

    addPolynomials(a, negB);
}

void multiplyPolynomials(Polynomial& a, Polynomial& b) {
    Polynomial result;
    Term* termA = a.head->next;

    while (termA != a.head) {
        Term* termB = b.head->next;

        while (termB != b.head) {
            int coef = termA->coef * termB->coef;
            int exp = termA->exp + termB->exp;
            result.insertTerm(coef, exp);
            termB = termB->next;
        }

        termA = termA->next;
    }

    *this = result;
}
```

```
    float evaluatePolynomial(float x) {
        float result = 0;
        Term* current = head->next;

        while (current != head) {
            result += current->coef * pow(x, current->exp);
            current = current->next;
        }

        return result;
    }

    void eraseTerm(int exponent) {
        Term* current = head->next;
        Term* prev = head;

        while (current != head) {
            if (current->exp == exponent) {
                prev->next = current->next;
                delete current;
                current = prev->next;
            } else {
                prev = current;
                current = current->next;
            }
        }
    }
};

int main() {
    Polynomial polyA, polyB, result;

    cout << "Enter Polynomial A:" << endl;
    polyA.readPolynomial();

    cout << "Enter Polynomial B:" << endl;
    polyB.readPolynomial();

    cout << "Polynomial A: ";
    polyA.displayPolynomial();
    cout << "Polynomial B: ";
    polyB.displayPolynomial();

    cout << "Adding A and B: ";
    result.addPolynomials(polyA, polyB);
    result.displayPolynomial();

    cout << "Subtracting B from A: ";
```

```
    result.subtractPolynomials(polyA, polyB);
    result.displayPolynomial();

    cout << "Multiplying A and B: ";
    result.multiplyPolynomials(polyA, polyB);
    result.displayPolynomial();

    float evalPoint;
    cout << "Enter a point to evaluate A: ";
    cin >> evalPoint;
    cout << "A(" << evalPoint << ") = " << polyA.evaluatePolynomial(evalPoint) << endl;

    int exp;
    cout << "Enter an exponent to erase from A: ";
    cin >> exp;
    polyA.eraseTerm(exp);
    cout << "A after erasing term with exponent " << exp << ": ";
    polyA.displayPolynomial();

    return 0;
}
```