Lab Assignment: Find the Sum of Two Sparse Polynomials Using Arrays

Objective

To understand the representation of sparse polynomials using arrays and perform polynomial addition.

Problem Statement

Given two sparse polynomials represented using arrays, compute and return their sum. The result should also be in sparse format.

Function Signature

void addPolynomials(Polynomial p1[], int n1, Polynomial p2[], int n2, Polynomial result[], int *nResult);

Where:

- `p1[]` and `p2[]` are arrays of terms representing two sparse polynomials.
- `n1` and `n2` are the number of terms in each polynomial respectively.
- `result[]` is the array to store the sum of the polynomials.
- `*nResult` will hold the number of terms in the resulting polynomial.

Each term of the polynomial can be represented using a structure:

```
typedef struct {
    int coeff;
    int exp;
} Polynomial;
```

Expected Behavior:

- Terms are added only when exponents match.
- Result should not include zero-coefficient terms.
- Terms in all polynomials are assumed to be in descending order of exponent.

Example:

Input:

Polynomial 1: `3x^4 + 2x^2 + 1`
Polynomial 2: `5x^3 + 2x^2 + 4`

Representation:

```
p1 = [{3,4}, {2,2}, {1,0}]
p2 = [{5,3}, {2,2}, {4,0}]
```

Output:

Sum: `3x^4 + 5x^3 + 4x^2 + 5`

```
result = [{3,4}, {5,3}, {4,2}, {5,0}]
```

Assumptions:

- Maximum number of terms in a polynomial is within array bounds.
- Duplicate exponents are not present in input polynomials.
- Input polynomials are sorted in descending order of exponents.

Algorithm:

1. Initialize three indices: `i = 0`, `j = 0`, `k = 0`.
2. Compare the exponents of `p1[i]` and `p2[j]`.
    - If `p1[i].exp > p2[j].exp`:
        - Copy `p1[i]` to `result[k]`
        - Increment `i`, `k`
    - Else if `p1[i].exp < p2[j].exp`:
        - Copy `p2[j]` to `result[k]`
        - Increment `j`, `k`
    - Else:
        - Add the coefficients
        - If the sum is non-zero, store in `result[k]` with the common exponent
        - Increment `i`, `j`, and possibly `k`
3. Copy any remaining terms from `p1` or `p2` into `result`.
4. Update `*nResult` with the total number of terms in `result`.